Cloud Computing RICS tutorial

Dan C. Marinescu

Computer Science Division

EECS Department, UCF

Email:dcm@cs.ucf.edu

The tutorial is based on the book

Cloud Computing: Theory and Practice

ISBN-13: **978-0124046276**
Published by Morgan Kaufmann in May-June 2013

http://www.amazon.com/Cloud-Computing-Practice-Dan-Marinescu/dp/0124046274/ref=sr_1_4?s=books&ie=UTF8&qid=1365357500&sr=1-4&keywords=Dan+C.+Marinescu

Contents

1. Basic concepts
2. Cloud computing infrastructure
3. Cloud applications
4. Virtualization
5. Resource management
6. Security

1. Basic concepts

- Network centric computing and network centric content.
- Cloud computing: the good, challenges, and vulnerabilities.
- Types of clouds.
- Cloud delivery models.

Network-centric computing

- Information processing can be done more efficiently on large farms of computing and storage systems accessible via the Internet.
  - Grid computing – initiated by the National Labs in the early 1990s; targeted primarily at scientific computing
  - Utility computing – initiated in 2005-2006 by IT companies and targeted at enterprise computing.

- The focus of utility computing is on the business model for providing computing services; it often requires a cloud-like infrastructure.

- Cloud computing is a path to utility computing embraced by major IT companies including: Amazon, HP, IBM, Microsoft, Oracle, and others.

Network-centric content

- Content: any type or volume of media, be it static or dynamic, monolithic or modular, live or stored, produced by aggregation, or mixed.

- The “Future Internet” will be content-centric; the creation and consumption of audio and visual content is likely to transform the Internet to support increased quality in terms of resolution, frame rate, color depth, stereoscopic information.

Network-centric computing and content

- Data-intensive: large scale simulation in science and engineering require large volumes of data. Multimedia streaming transfers large volume of data.
- Network-intensive: transferring large volumes of data requires high bandwidth networks
- Low-latency networks for data streaming, parallel computing, computation steering.
- The systems are accessed using *thin clients* running on systems with limited resources, e.g., wireless devices such as smart phones and tablets.
- The infrastructure should support some form of workflow management.

Evolution of concepts and technologies

- The web and the semantic web - expected to support composition of services. The web is dominated by unstructured or semi-structured data, while the semantic web advocates inclusion of sematic content in web pages.
- The Grid - initiated in the early 1990s by National Laboratories and Universities; used primarily for applications in the area of science and engineering.
- Peer-to-peer systems
- Computer clouds

Cloud computing

- Uses Internet technologies to offer scalable and elastic services. The term “elastic computing refers to the ability of *dynamically acquiring computing resources* and supporting a variable workload.
- The resources used for these services can be metered and the *users can be charged only for the resources they used*.
- The maintenance and security are ensured by service providers.
- The service providers can operate more efficiently due to specialization and centralization.

Cloud computing (cont'd)

- Lower costs for the cloud service provider are past to the cloud users.

- Data is stored:
  - closer to the site where it is used.
  - in a device and in a location-independent manner.

- The data storage strategy can increases reliability, as well as security and lower communication costs

Types of clouds

- Public Cloud - the infrastructure is made available to the general public or a large industry group and is owned by the organization selling cloud services.
- Private Cloud - infrastructure operated solely for an organization.
- Community Cloud - the infrastructure is shared by several organizations and supports a specific community that has shared.
- Hybrid Cloud - composition of two or more clouds (public, private, or community) bound by standardized technology that enables data and application portability.

The “good” about cloud computing

- Resources such as CPU cycles, storage, network bandwidth are shared.
- When multiple applications share a system their peak demands for resources are not synchronized thus, *multiplexing leads to a higher resource utilization*.
- Resources can be aggregated to support data-intensive applications.
- Data sharing facilitates collaborative activities. Many applications require multiple types of analysis of shared data sets and multiple decisions carried out by groups scattered around the globe.

More “good” about cloud computing

- Eliminate the initial investment costs for a private computing infrastructure and the maintenance and operation costs.
- Cost reduction: concentration of resources creates the opportunity to pay as you go for computing and thus
- Elasticity: the ability to accommodate workloads with very large peak-to-average ratios.
- User convenience: virtualization allows users to operate in familiar environments rather than in idiosyncratic ones.

Why cloud computing could be successful when other paradigms have failed?

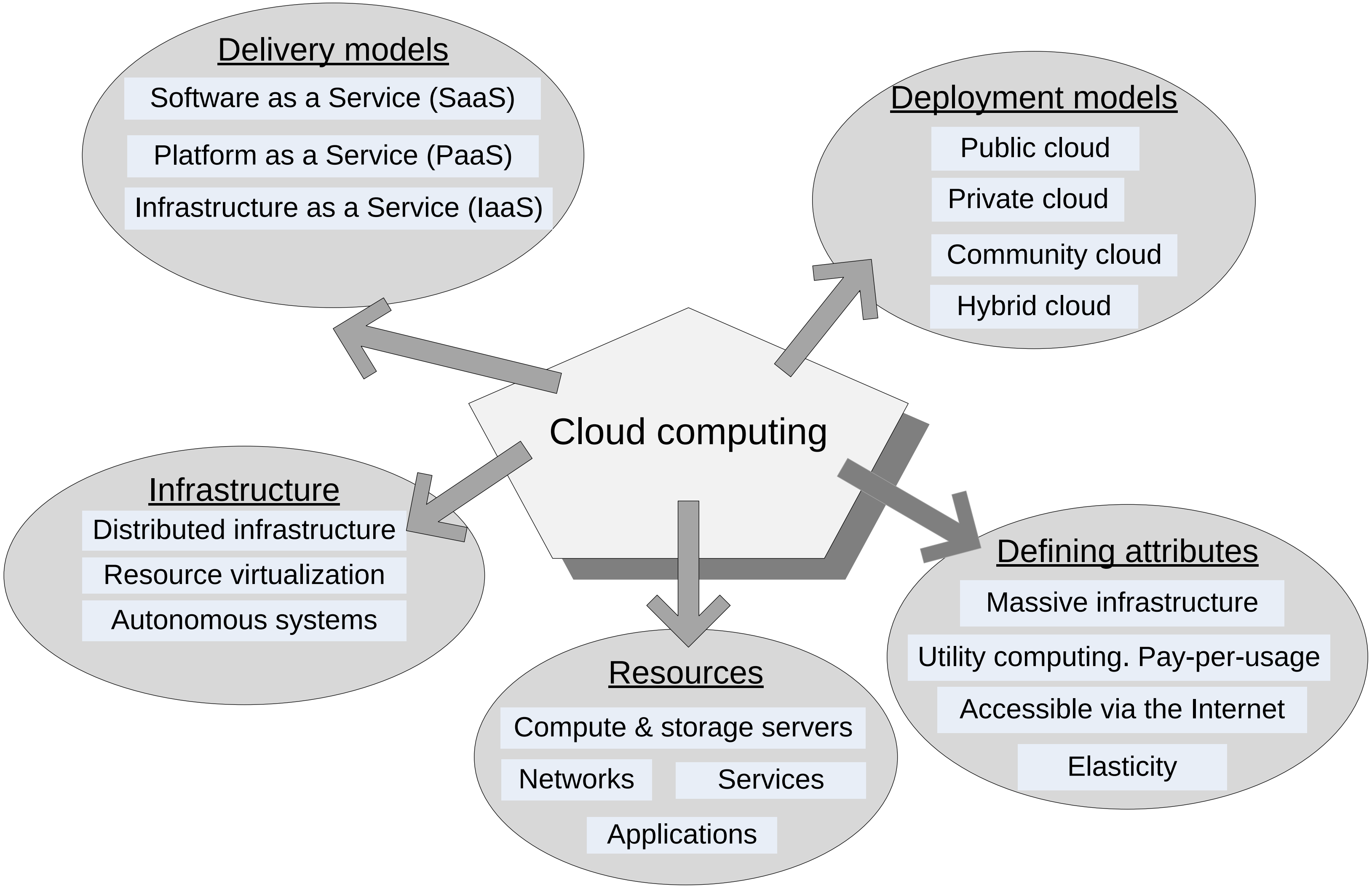
- It is in a better position to exploit recent advances in software, networking, storage, and processor technologies promoted by the same companies who provide cloud services.
- It is focused on enterprise computing; its adoption by industrial organizations, financial institutions, government, and so on could have a huge impact on the economy.
- A cloud consists of a homogeneous set of hardware and software resources.
- The resources are in a single administrative domain (AD). Security, resource management, fault-tolerance, and quality of service are less challenging than in a heterogeneous environment with resources in multiple ADs.

Challenges for cloud computing

- Availability of service; what happens when the service provider cannot deliver?
- Diversity of services, data organization, user interfaces available at different service providers limit user mobility; once a customer is hooked to one provider it is hard to move to another. Standardization efforts at NIST!
- Data confidentiality and auditability, a serious problem.
- Data transfer bottleneck; many applications are data-intensive.

More challenges

- Performance unpredictability, one of the consequences of resource sharing.
  - How to use resource virtualization and performance isolation for QoS guarantees?
  - How to support elasticity, the ability to scale up and down quickly?
- Resource management; is self-organization and self-management a solution?
- Security and confidentiality; major concern.
- Addressing these challenges provides good research opportunities!!


Cloud computing
Delivery models
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Deployment models
Public cloud
Private cloud
Community cloud
Hybrid cloud
Infrastructure
Distributed infrastructure
Resource virtualization
Autonomous systems
Resources
Compute & storage servers
Networks
Services
Applications
Defining attributes
Massive infrastructure
Utility computing. Pay-per-usage
Accessible via the Internet
Elasticity

Cloud delivery models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- In frastructure as a Service (IaaS)

Software as a Service (SaaS)

- Applications are supplied by the service provider.
- The user does not manage or control the underlying cloud infrastructure or individual application capabilities.
- Services offered include:
  - Enterprise services such as: workflow management, group-ware and collaborative, supply chain, communications, digital signature, customer relationship management (CRM), desktop software, financial management, geo-spatial, and search.
  - Web 2.0 applications such as: metadata management, social networking, blogs, wiki services, and portal services.
- Not suitable for real-time applications or those where data is not allowed to be hosted externally.
- Examples: Gmail, Google search engine.

Platform as a Service (PaaS)

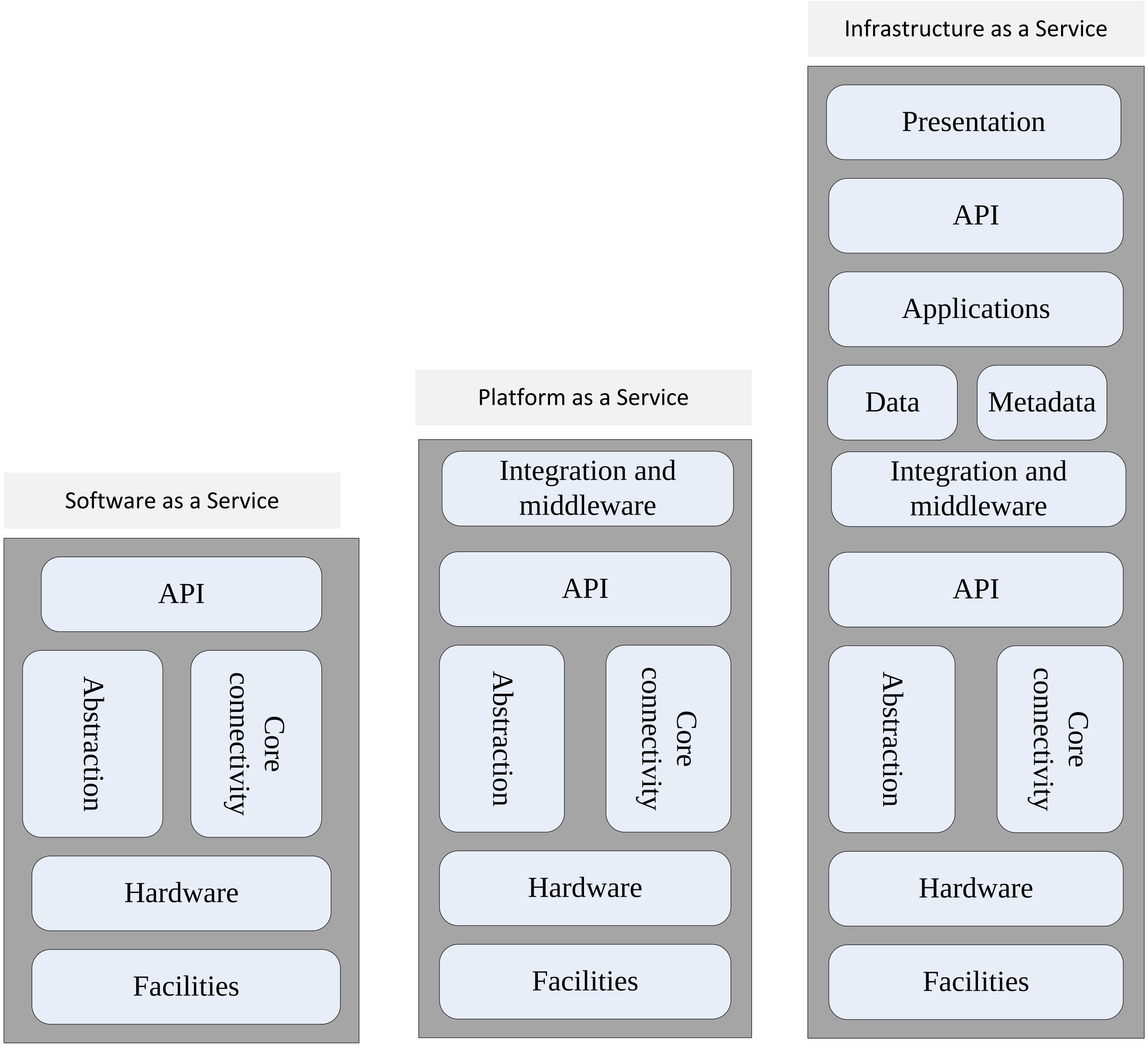
- Allows a cloud user to deploy consumer-created or acquired applications using programming languages and tools supported by the service provider.

- The user:
  - □ has control over the deployed applications and, possibly, application hosting environment configurations;
  - □ does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage.

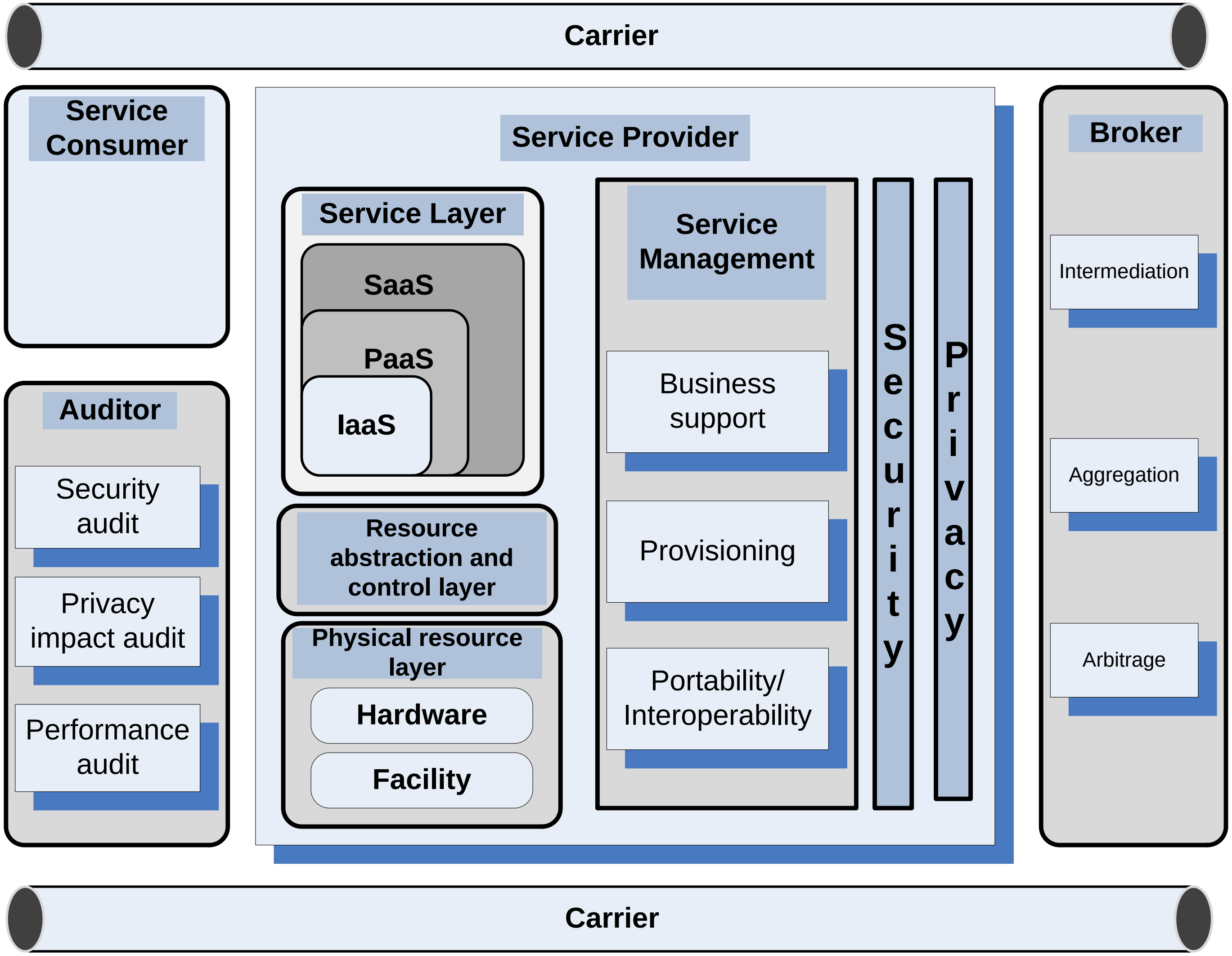
- Not particularly useful when:
  - □ the application must be portable;
  - □ proprietary programming languages are used;
  - □ the hardware and software must be customized to improve the performance of the application.

Infrastructure as a Service (IaaS)

- The user is able to deploy and run arbitrary software, which can include operating systems and applications.
- The user does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of some networking components, e.g., host firewalls.
- Services offered by this delivery model include: server hosting, web servers, storage, computing hardware, operating systems, virtual instances, load balancing, Internet access, and bandwidth provisioning.


Infrastructure as a Service
Presentation
API
Applications
Data
Metadata
Integration and middleware
API
Abstraction
Core connectivity
Hardware
Facilities
Platform as a Service
Integration and middleware
API
Abstraction
Core connectivity
Hardware
Facilities
Software as a Service
API
Abstraction
Core connectivity
Hardware
Facilities

NIST cloud reference model



Ethical issues

- Paradigm shift with implications on computing ethics:
  - the control is relinquished to third party services;
  - the data is stored on multiple sites administered by several organizations;
  - multiple services interoperate across the network.
- Implications
  - unauthorized access;
  - data corruption;
  - infrastructure failure, and service unavailability.

De-perimeterisation

- Systems can span the boundaries of multiple organizations and cross the security borders.
- The complex structure of cloud services can make it difficult to determine who is responsible in case something undesirable happens.
- Identity fraud and theft are made possible by the unauthorized access to personal data in circulation and by new forms of dissemination through social networks and they could also pose a danger to cloud computing.

Privacy issues

- Cloud service providers have already collected petabytes of sensitive personal information stored in data centers around the world. The acceptance of cloud computing therefore will be determined by privacy issues addressed by these companies and the countries where the data centers are located.

- Privacy is affected by cultural differences; some cultures favor privacy, others emphasize community. This leads to an ambivalent attitude towards privacy in the Internet which is a global system.

Cloud vulnerabilities

- Clouds are affected by malicious attacks and failures of the infrastructure, e.g., power failures.

- Such events can affect the Internet domain name servers and prevent access to a cloud or can directly affect the clouds
  - in 2004 an attack at Akamai caused a domain name outage and a major blackout that affected Google, Yahoo, and other sites.
  - in 2009, Google was the target of a denial of service attack which took down Google News and Gmail for several days;
  - in 2012 lightning caused a prolonged down time at Amazon.

2. Cloud infrastructure

- IaaS services from Amazon
- Open-source platforms for private clouds
- Cloud storage diversity and vendor lock-in
- Cloud interoperability; the Intercloud
- Energy use and ecological impact large datacenters
- Service and compliance level agreements
- Responsibility sharing between user and the cloud service provider

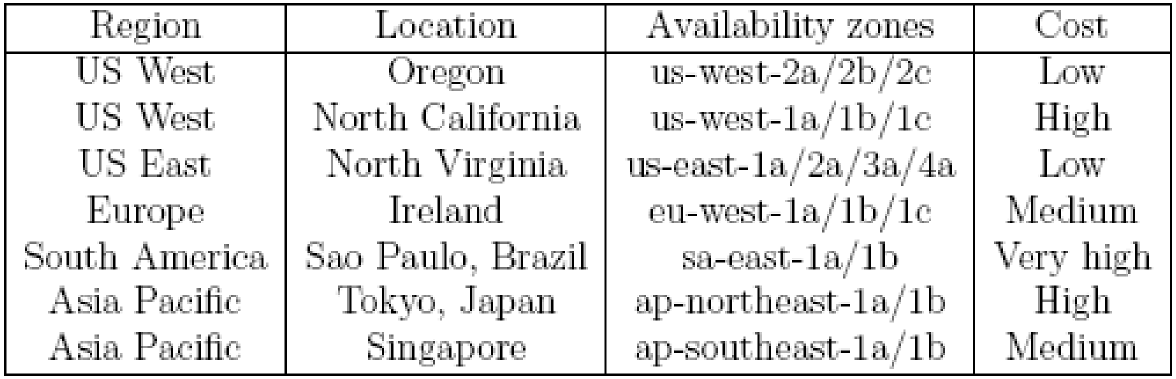
Existing cloud infrastructure

- The cloud computing infrastructure at Amazon, Google, and Microsoft (as of mid 2012).
  - Amazon is a pioneer in Infrastructure-as-a-Service (IaaS)
  - Google's efforts are focused on Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS)
  - Microsoft is involved in PaaS
- Private clouds are an alternative to public clouds. Open-source cloud computing platforms such as
  - Eucalyptus
  - OpenNebula
  - Nimbus
  - OpenStack

  can be used as a control infrastructure for a private cloud.

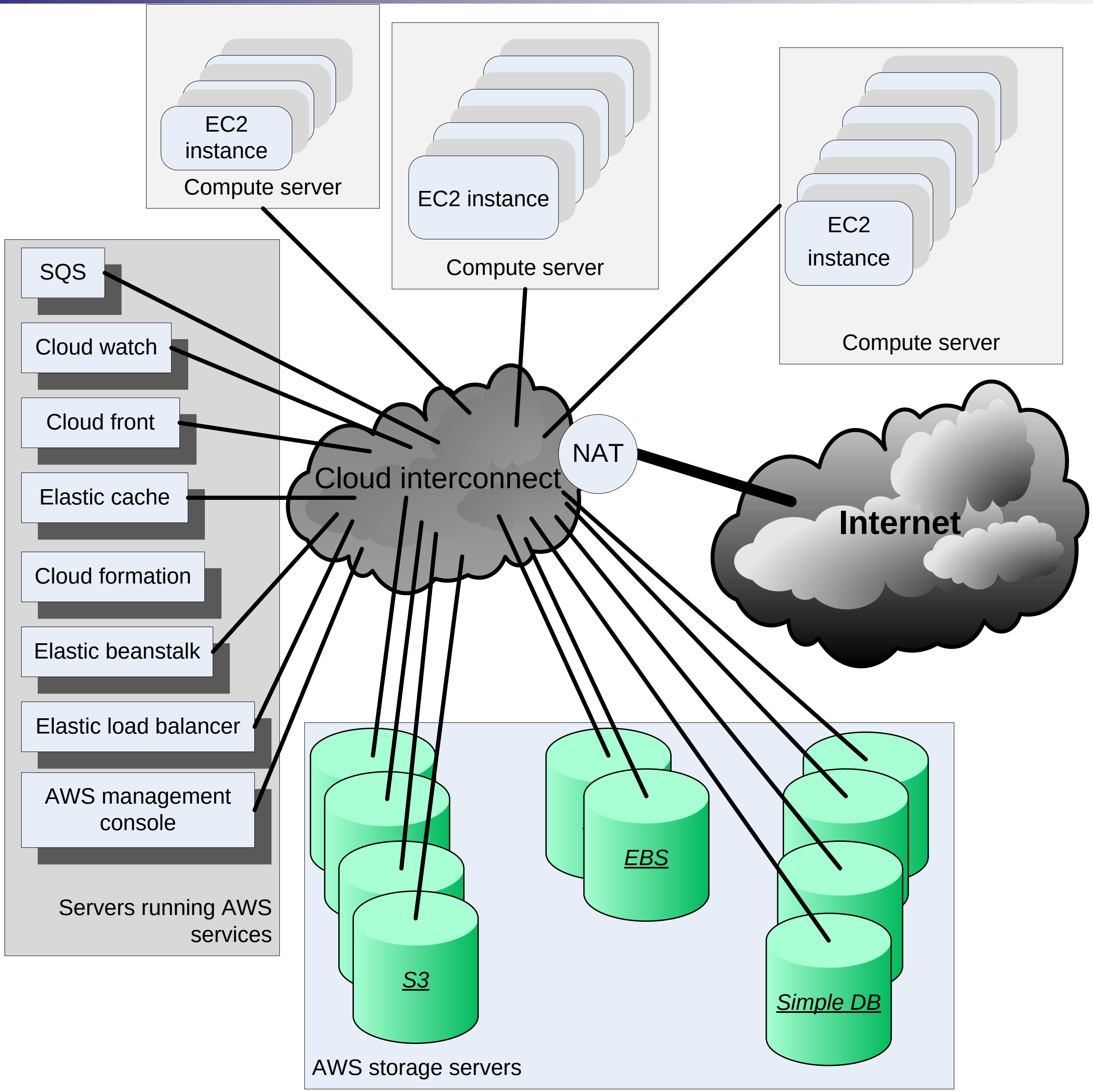
AWS regions and availability zones

- Amazon offers cloud services through a network of data centers on several continents.
- In each *region* there are several availability zones interconnected by high-speed networks.
- An *availability zone* is a data center consisting of a large number of servers.



Region	Location	Availability zones	Cost
US West	Oregon	us-west-2a/2b/2c	Low
US West	North California	us-west-1a/1b/1c	High
US East	North Virginia	us-east-1a/2a/3a/4a	Low
Europe	Ireland	eu-west-1a/1b/1c	Medium
South America	Sao Paulo, Brazil	sa-east-1a/1b	Very high
Asia Pacific	Tokyo, Japan	ap-northeast-1a/1b	High
Asia Pacific	Singapore	ap-southeast-1a/1b	Medium

- Regions do not share resources and communicate through the Internet.

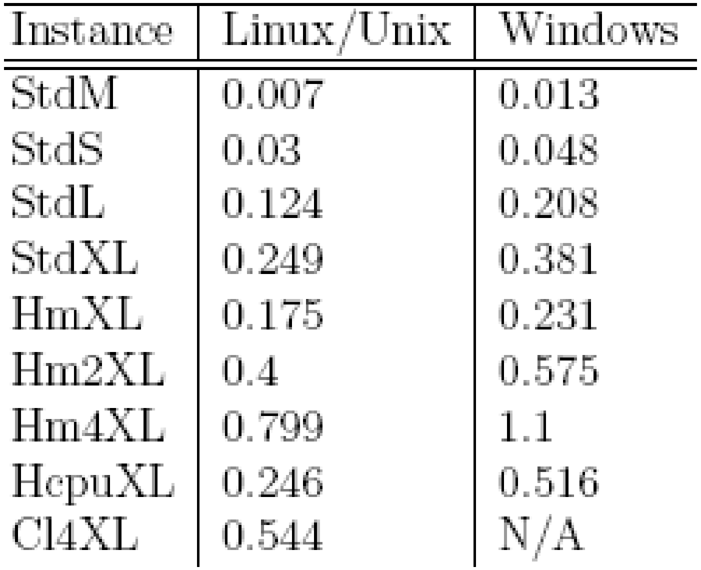

EC2 instance
Compute server
EC2 instance
Compute server
EC2 instance
Compute server
SQS
Cloud watch
Cloud front
Elastic cache
Cloud formation
Elastic beanstalk
Elastic load balancer
AWS management console
Servers running AWS services
Cloud interconnect
NAT
Internet
S3
EBS
Simple DB
AWS storage servers

Steps to run an application

- Retrieve the user input from the front-end.
- Retrieve the disk image of a VM (Virtual Machine) from a repository.
- Locate a system and requests the VMM (Virtual Machine Monitor) running on that system to setup a VM.
- Invoke the Dynamic Host Configuration Protocol (DHCP) and the IP bridging software to set up a MAC and IP address for the VM.

Instance cost

- A main attraction of the Amazon cloud computing is the low cost.



Instance	Linux/Unix	Windows
StdM	0.007	0.013
StdS	0.03	0.048
StdL	0.124	0.208
StdXL	0.249	0.381
HmXL	0.175	0.231
Hm2XL	0.4	0.575
Hm4XL	0.799	1.1
HcpuXL	0.246	0.516
Cl4XL	0.544	N/A

AWS services introduced in 2012

- *Route 53* - low-latency DNS service used to manage user's DNS public records.
- *Elastic MapReduce (EMR)* - supports processing of large amounts of data using a hosted Hadoop running on EC2.
- *Simple Workflow Service (SWF)* - supports workflow management; allows scheduling, management of dependencies, and coordination of multiple EC2 instances.
- *ElastiCache* - enables web applications to retrieve data from a managed in-memory caching system rather than a much slower disk-based database.
- *DynamoDB* - scalable and low-latency fully managed NoSQL database service;

AWS services introduced in 2012 (cont'd)

- *CloudFront* - web service for content delivery.
- *Elastic Load Balancer* - automatically distributes the incoming requests across multiple instances of the application.
- *Elastic Beanstalk* - handles automatically deployment, capacity provisioning, load balancing, auto-scaling, and application monitoring functions.
- *CloudFormation* - allows the creation of a stack describing the infrastructure for an application.

Elastic Beanstalk

- Handles automatically the deployment, capacity provisioning, load balancing, auto-scaling, and monitoring functions.
- Interacts with other services including EC2, S3, SNS, Elastic Load Balance and AutoScaling.
- The management functions provided by the service are:
  - deploy a new application version (or rollback to a previous version);
  - access to the results reported by CloudWatch monitoring service;
  - email notifications when application status changes or application servers are added or removed; and
  - access to server log files without needing to login to the application servers.
- The service is available using: a Java platform, the PHP server-side description language, or the .NET framework.

Open-source platforms for private clouds

- Eucalyptus - can be regarded as an open-source counterpart of Amazon's EC2.

- *Open-Nebula* - a private cloud with users actually logging into the head node to access cloud functions. The system is centralized and its default configuration uses the NFS filesystem.

- *Nimbus* - a cloud solution for scientific applications based on Globus software; inherits from Globus
  - the image storage,
  - the credentials for user authentication,
  - the requirement that a running Nimbus process can ssh into all compute nodes.

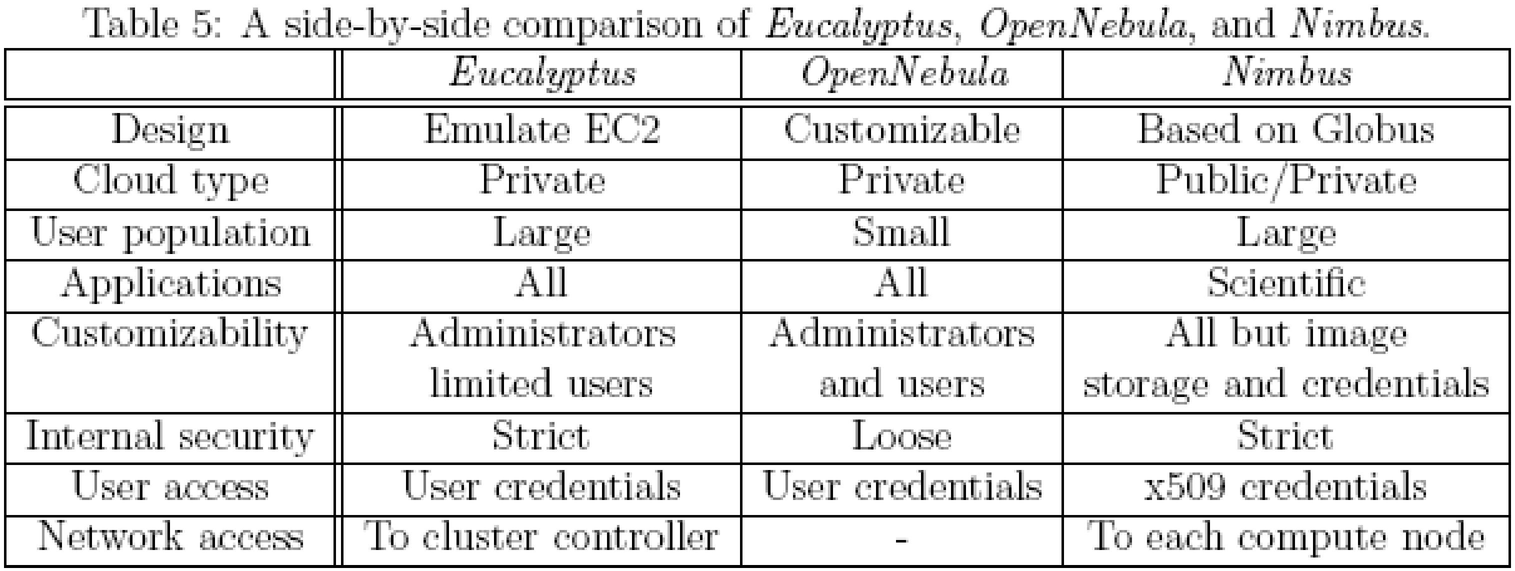
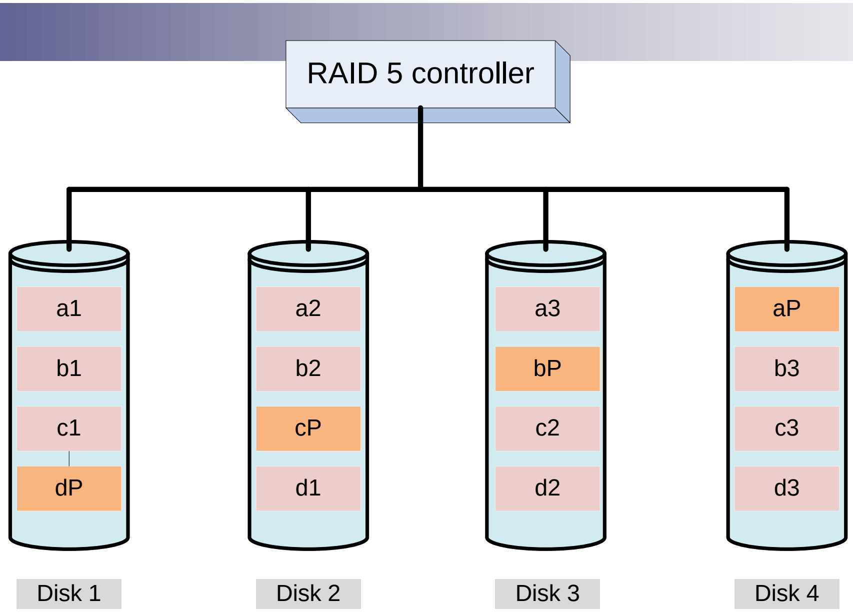


Table 5: A side-by-side comparison of *Eucalyptus*, *OpenNebula*, and *Nimbus*.

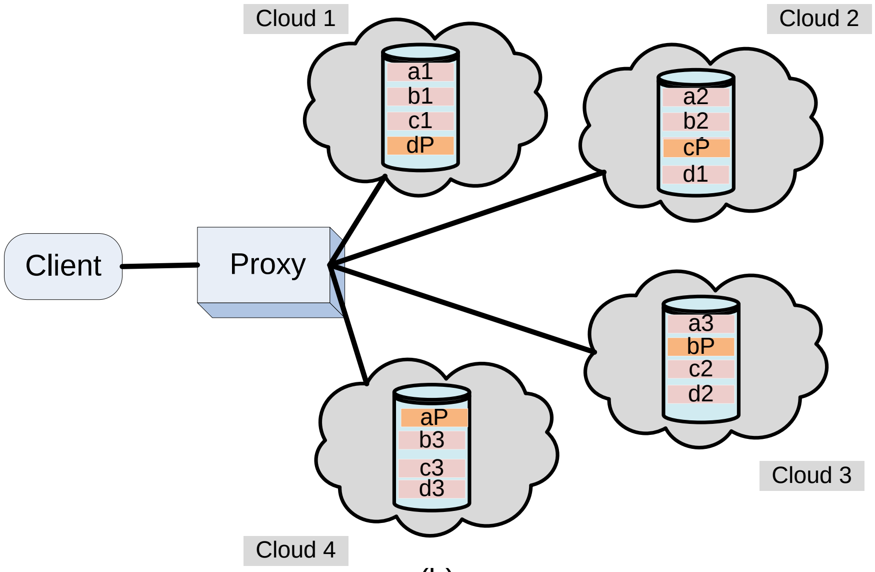
	Eucalyptus	*OpenNebula*	*Nimbus*
Design	Emulate EC2	Customizable	Based on Globus
Cloud type	Private	Private	Public/Private
User population	Large	Small	Large
Applications	All	All	Scientific
Customizability	Administrators limited users	Administrators and users	All but image storage and credentials
Internal security	Strict	Loose	Strict
User access	User credentials	User credentials	x509 credentials
Network access	To cluster controller	-	To each compute node

Cloud storage diversity and vendor lock-in

- Risks when a large organization relies on a single cloud service provider:
  - cloud services may be unavailable for a short, or an extended period of time;
  - permanent data loss in case of a catastrophic system failure;
  - the provider may increase the prices for service.
- Switching to another provider could be very costly due to the large volume of data to be transferred from the old to the new provider.
- A solution is to replicate the data to multiple cloud service providers, similar to data replication in RAID.


RAID 5 controller
a1
b1
c1
dP
a2
b2
cP
d1
a3
bP
c2
d2
aP
b3
c3
d3
Disk 1
Disk 2
Disk 3
Disk 4

(a)


Cloud 1
a1
b1
c1
dP
Cloud 2
a2
b2
cP
d1
Client
Proxy
Cloud 3
a3
bP
c2
d2
Cloud 4
aP
b3
c3
d3

(b)

Cloud interoperability; the Intercloud

- Is an Intercloud, a federation of clouds that cooperate to provide a better user experience feasible?
- Not likely at this time:
  - there are no standards for either storage of processing;
  - the clouds are based on different delivery models;
  - the set of services supported by these delivery models is large and open; new services are offered every few months;
  - CSPs (Cloud Service Providers) belive that they have a competitive advantage due to the uniqueness of the added value of their services;
  - Security is a major concern for cloud users and an Intercloud could only create new threats.

Energy use and ecological impact

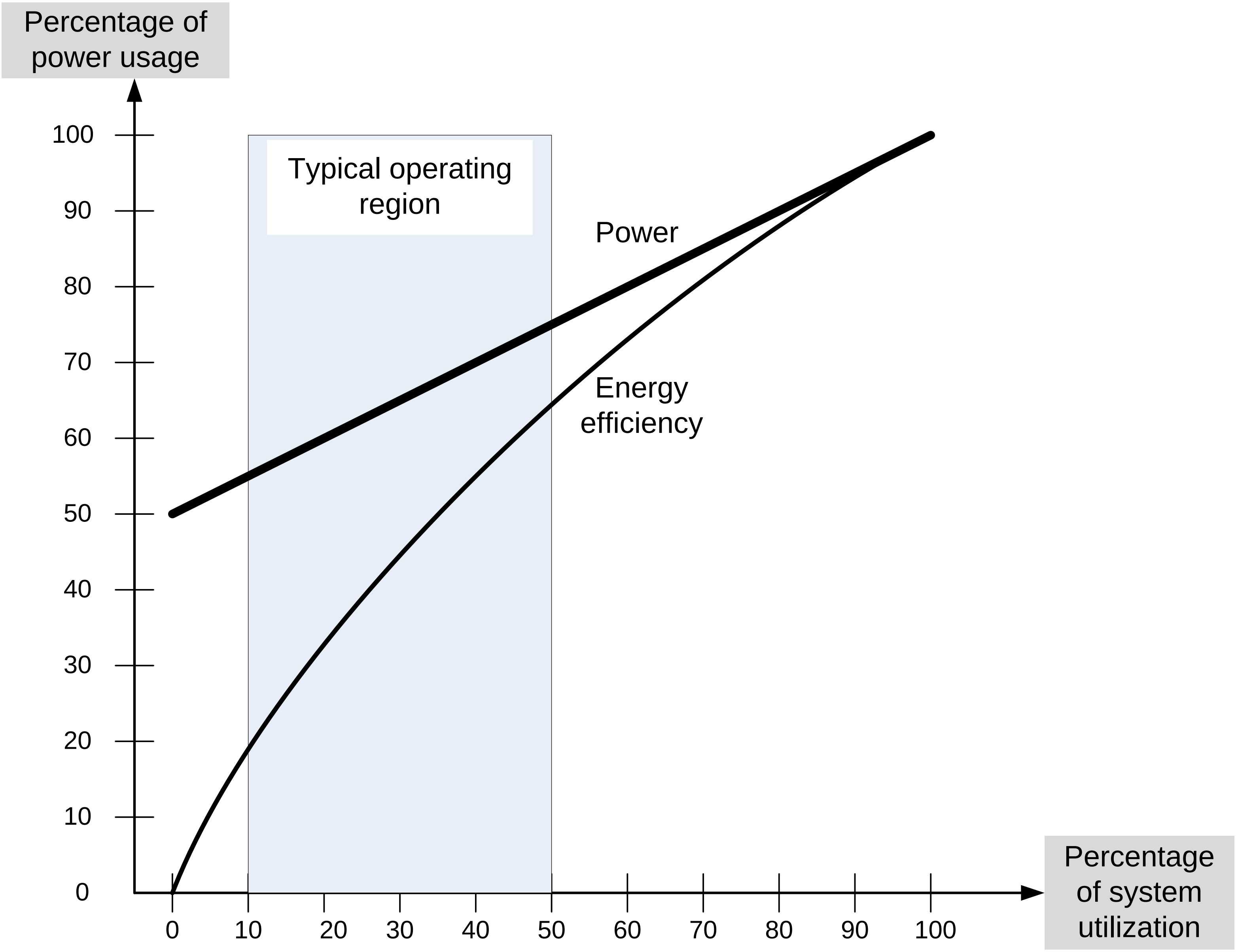
- The energy consumption of large-scale data centers and their costs for energy and for cooling are significant.
- In 2006, the 6,000 data centers in the U.S consumed $61x10^9$ KWh of energy, 1.5% of all electricity consumption, at a cost of $4.5 billion.
- The energy consumed by the data centers was expected to double from 2006 to 2011 and peak instantaneous demand to increase from 7 GW to 12 GW.
- The greenhouse gas emission due to the data centers is estimated to increase from $116 \, x10^9$ tones of CO_2 in 2007 to 257 tones in 2020 due to increased consumer demand.
- The effort to reduce energy use is focused on computing, networking, and storage activities of a data center.

Energy use and ecological impact (cont'd)

- Operating efficiency of a system is captured by the *performance per Watt of power*.
- The performance of supercomputers has increased 3.5 times faster than their operating efficiency - 7000% versus 2,000% during the period 1998 – 2007.
- A typical Google cluster spends most of its time within the 10-50% CPU utilization range; there is a mismatch between server workload profile and server energy efficiency.

Energy-proportional systems

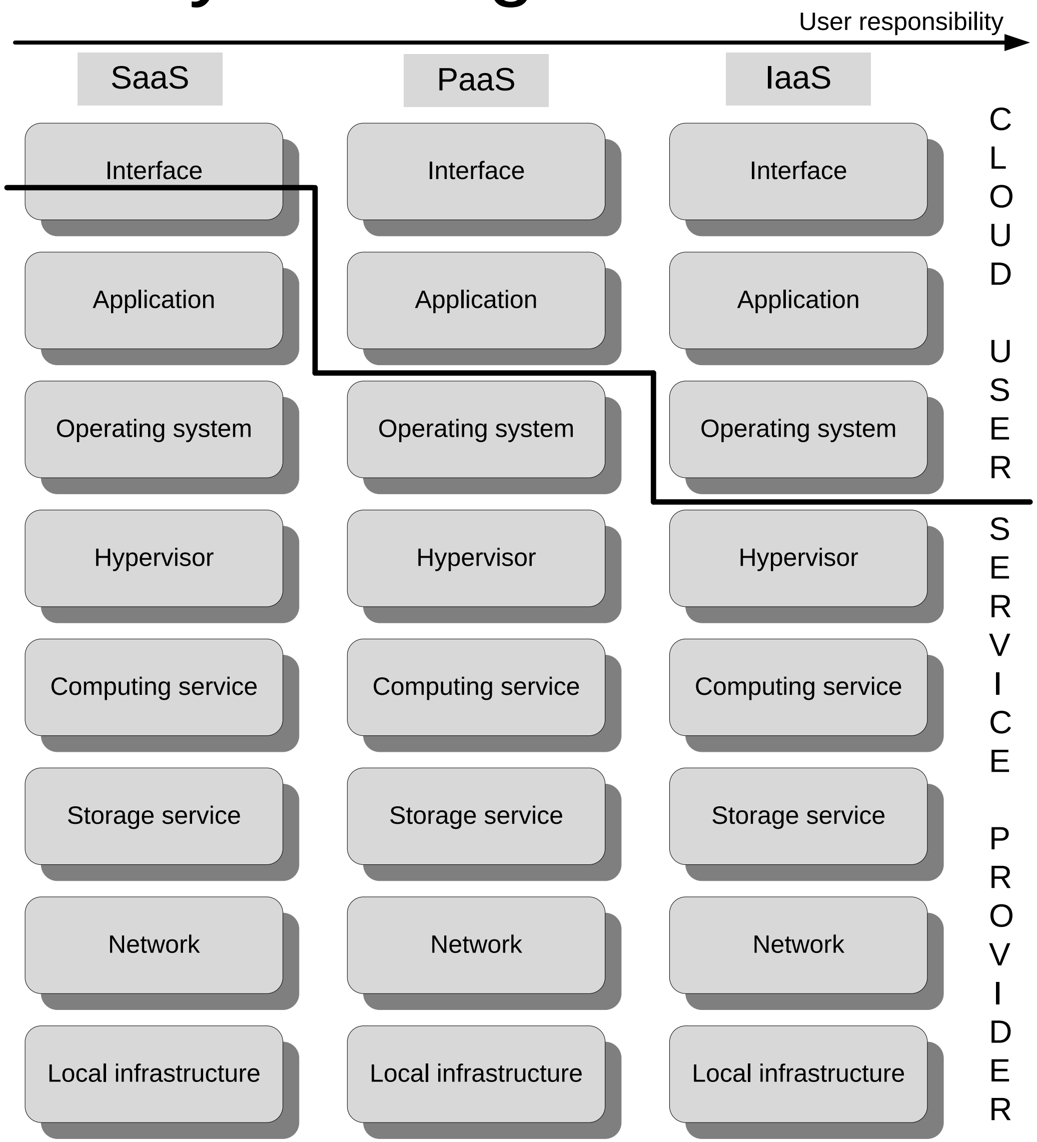
- An energy-proportional system consumes no power when idle, very little power under a light load and, gradually, more power as the load increases.
- By definition, an ideal energy-proportional system is always operating at 100% efficiency.
- Humans are a good approximation of an ideal energy proportional system; about 70 W at rest, 120 W on average on a daily basis, and can go as high as 1,000 – 2,000 W during a strenuous, short time effort.
- Even when power requirements scale linearly with the load, the energy efficiency of a computing system is not a linear function of the load; even when idle, a system may use 50% of the power corresponding to the full load


Percentage of power usage
100
90
80
70
60
50
40
30
20
10
0
Typical operating region
Power
Energy efficiency
0
10
20
30
40
50
60
70
80
90
100
Percentage of system utilization

Service Level Agreement (SLA)

- SLA - a negotiated contract between the customer and CSP; can be legally binding or informal. Objectives:
  - Identify and define the customer's needs and constraints including the level of resources, security, timing, and QoS.
  - Provide a framework for understanding; a critical aspect of this framework is a clear definition of classes of service and the costs.
  - Simplify complex issues; clarify the boundaries between the responsibilities of clients and CSP in case of failures.
  - Reduce areas of conflict.
  - Encourage dialog in the event of disputes.
  - Eliminate unrealistic expectations.
- Specifies the services that the customer receives, rather than how the cloud service provider delivers the services.

Responsibility sharing between user and CSP



User responsibility

SaaS	PaaS	IaaS
Interface	Interface	Interface
Application	Application	Application
Operating system	Operating system	Operating system
Hypervisor	Hypervisor	Hypervisor
Computing service	Computing service	Computing service
Storage service	Storage service	Storage service
Network	Network	Network
Local infrastructure	Local infrastructure	Local infrastructure

CLOUD USER

SERVICE PROVIDER

User security concerns

- Potential loss of control/ownership of data.
- Data integration, privacy enforcement, data encryption.
- Data remanence after de-provisioning.
- Multi tenant data isolation.
- Data location requirements within national borders.
- Hypervisor security.
- Audit data integrity protection.
- Verification of subscriber policies through provider controls.
- Certification/Accreditation requirements for a given cloud service.

3. Cloud applications and paradigms

- Existing cloud applications and new opportunities
- Architectural styles for cloud applications
- Coordination based on a state machine model – the Zookeeper
- The MapReduce programming model
- Clouds for science and engineering
- High performance computing on a cloud
- Legacy applications on a cloud
- Social computing, digital content, and cloud computing

Cloud applications

- Cloud computing is very attractive to the users:
  - Economic reasons
    - low infrastructure investment
    - low cost - customers are only billed for resources used
  - Convenience and performance
    - application developers enjoy the advantages of a just-in-time infrastructure they are free to design an application without being concerned with the system where the application will run;
    - the potential to reduce the execution time of compute-intensive and data-intensive applications through parallelization. If an application can partition the workload in n segments and spawn *n* instances of itself, then the execution time could be reduced by a factor close to *n*.
- Cloud computing is also beneficial for the providers of computing cycles - it typically leads to a higher level of resource utilization.

Cloud applications (cont'd)

- Ideal applications for cloud computing:
  - Web services;
  - Database services;
  - Transaction-based services; The resource requirements of transaction-oriented services benefit from an elastic environment where resources are available when needed and where one pays only for the resources it consumes.
- Applications unlikely to perform well on a cloud:
  - Applications with a complex workflow and multiple dependencies, as is often the case in high-performance computing.
  - Applications which require intensive communication among concurrent instances.
  - When the workload cannot be arbitrarily partitioned.

Cloud application development

- Challenges
  - Performance isolation is nearly impossible to reach in a real system, especially when the system is heavily loaded.
  - Reliability - major concern; server failures expected when a large number of servers cooperate for the computations.
  - Cloud infrastructure exhibits latency and bandwidth fluctuations which affect the application performance.
  - Performance considerations limit the amount of *data logging*; the ability to identify the source of unexpected results and errors is helped by frequent logging.

Existing and new application opportunities

- Three broad categories of existing applications:
  - Processing pipelines;
  - Batch processing systems; and
  - Web applications
- Potentially new applications
  - Batch processing for decision support systems and business analytics
  - Mobile interactive applications which process large volumes of data from different types of sensors;
  - Science and engineering could greatly benefit from cloud computing as many applications in these areas are compute-intensive and data-intensive.

Processing pipelines

- Indexing large datasets created by web crawler engines.
- Data mining - searching large collections of records to locate items of interests.
- Image processing
  - image conversion, e.g., enlarge an image or create thumbnails;
  - compress or encrypt images.
- Video transcoding from one video format to another, e.g., from AVI to MPEG.
- Document processing;
  - convert large collection of documents from one format to another, e.g., from Word to PDF
  - encrypt the documents;
  - use Optical Character Recognition to produce digital images of documents.

Batch processing applications

- Generation of daily, weekly, monthly, and annual activity reports for retail, manufacturing, other economical sectors.
- Processing, aggregation, and summaries of daily transactions for financial institutions, insurance companies, and healthcare organizations.
- Processing billing and payroll records.
- Management of the software development, e.g., nightly updates of software repositories.
- Automatic testing and verification of software and hardware systems.

Web access

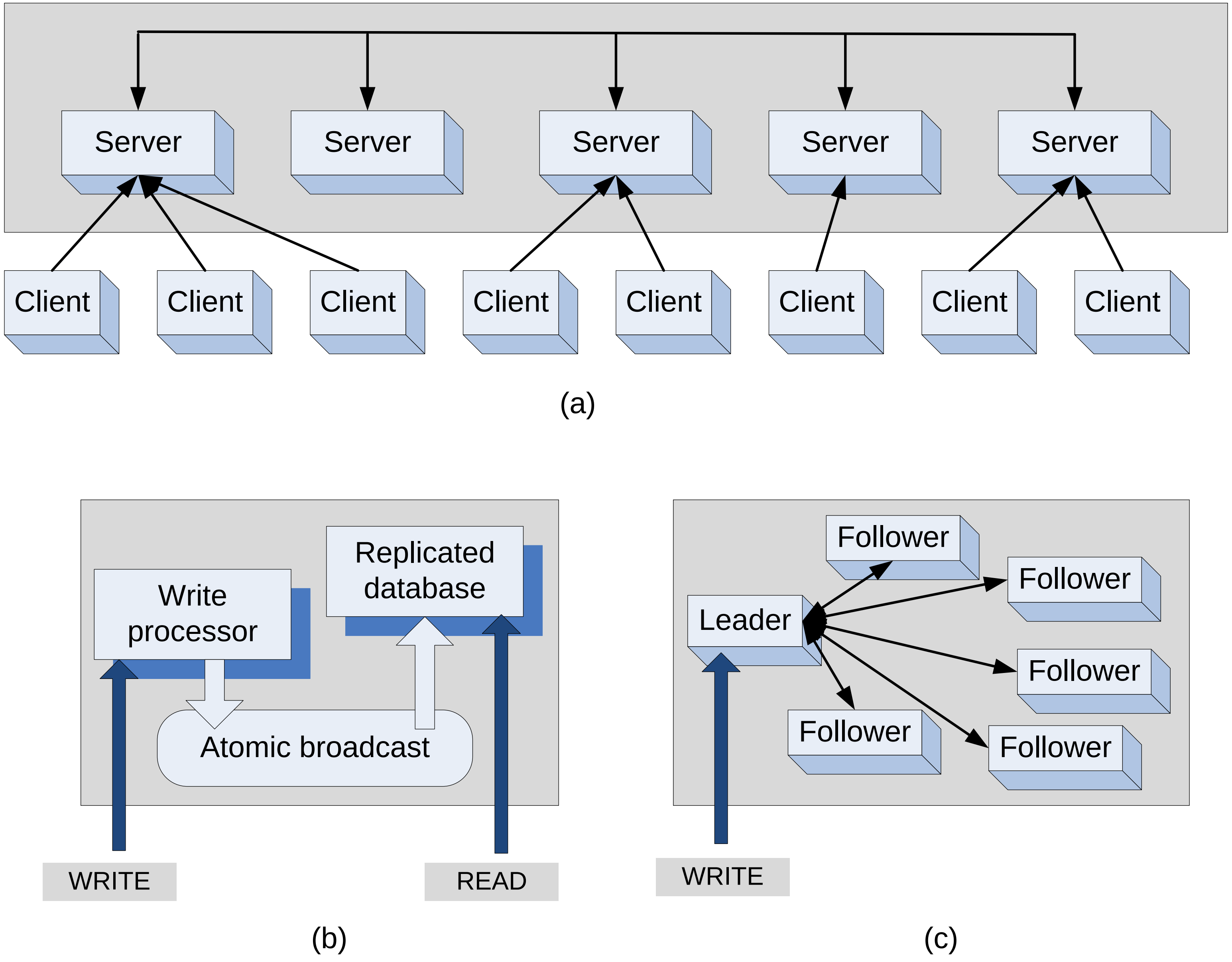
- Sites for online commerce
- Sites with a periodic or temporary presence.
  - Conferences or other events.
  - Active during a particular season (e.g., the Holidays Season) or income tax reporting.
- Sites for promotional activities
- Sites that ``sleep'' during the night and auto-scale during the day.

Architectural styles for cloud applications

- Based on the client-server paradigm.
- Stateless servers - view a client request as an independent transaction and respond to it; the client is not required to first establish a connection to the server.
- Often clients and servers communicate using Remote Procedure Calls (RPCs).
- Simple Object Access Protocol (SOAP) - application protocol for web applications; message format based on the XML. Uses TCP or UDP transport protocols.
- Representational State Transfer (REST) - software architecture for distributed hypermedia systems. Supports client communication with stateless servers, it is platform independent, language independent, supports data caching, and can be used in the presence of firewalls.

Coordination - ZooKeeper

- Cloud elasticity → distribute computations and data across multiple systems; coordination among these systems is a critical function in a distributed environment.
- ZooKeeper
  - distributed coordination service for large-scale distributed systems;
  - high throughput and low latency service;
  - implements a version of the Paxos consensus algorithm;
  - open-source software written in Java with bindings for Java and C.
  - the servers in the pack communicate and elect a leader;
  - a database is replicated on each server; consistency of the replicas is maintained;
  - a client connect to a single server, synchronizes its clock with the server, and sends requests, receives responses and watch events through a TCP connection.


Server
Server
Server
Server
Server
Client
Client
Client
Client
Client
Client
Client
Client
(a)
Write processor
Replicated database
Atomic broadcast
WRITE
READ
(b)
Follower
Follower
Leader
Follower
Follower
Follower
WRITE
(c)

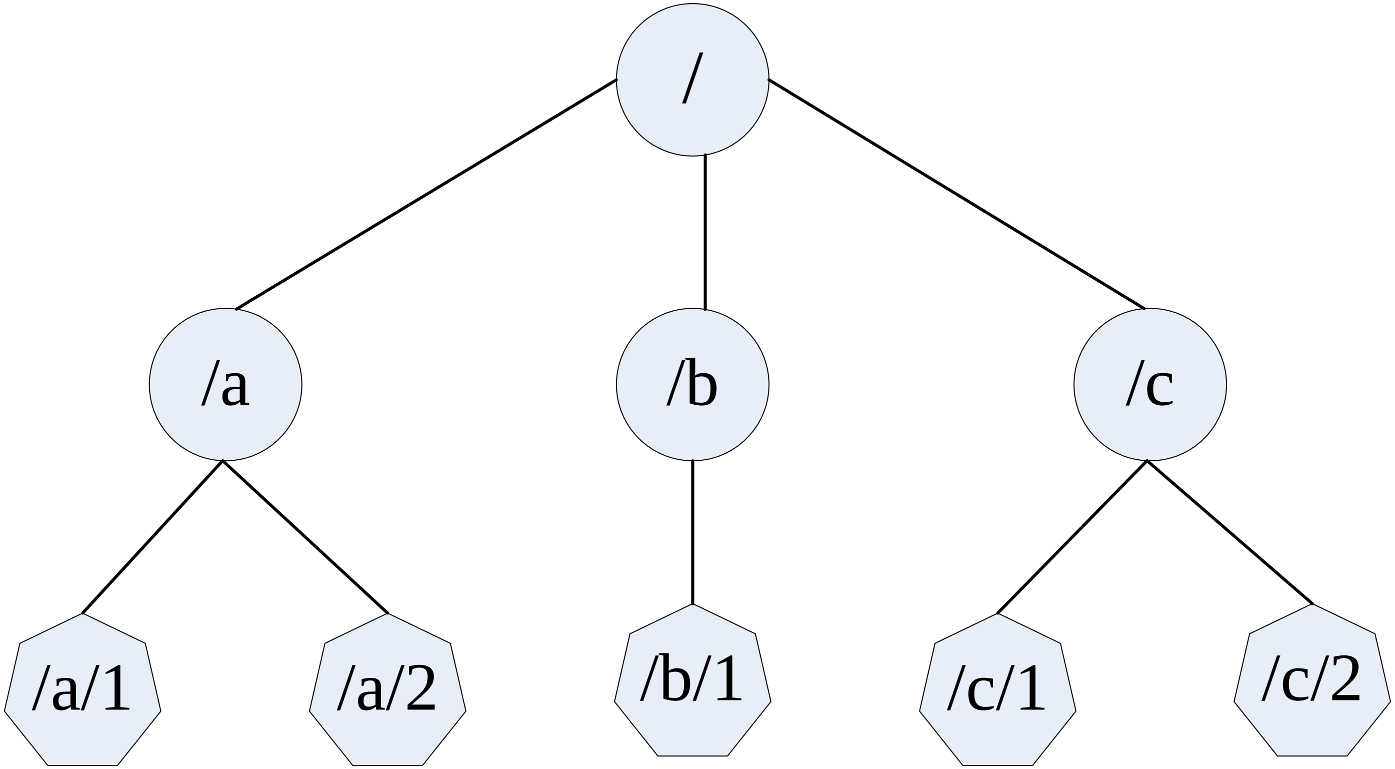
Zookeeper communication

- Messaging layer → responsible for the election of a new leader when the current leader fails.
- Messaging protocols uses:
  - packets - sequence of bytes sent through a FIFO channel,
  - proposals - units of agreement, and
  - messages - sequence of bytes atomically broadcast to all servers.
- A message is included into a proposal and it is agreed upon before it is delivered.
- Proposals are agreed upon by exchanging packets with a quorum of servers as required by the Paxos algorithm.

Zookeeper communication (cont'd)

- Messaging layer guarantees
  - Reliable delivery: if a message m is delivered to one server, it will be eventually delivered to all servers;
  - Total order: if message m a is delivered before message n to one server, a will be delivered before n to all servers;
  - Causal order: if message n is sent after m has been delivered by the sender of n, then m must be ordered before n.

Shared hierarchical namespace similar to a file system; znodes instead of inodes



ZooKeeper service guarantees

- The guarantees provided by Zookeeper:
  - Atomicity - a transaction either completes or fails.
  - Sequential consistency of updates - updates are applied strictly in the order they are received.
  - Single system image for the clients - a client receives the same response regardless of the server it connects to.
  - Persistence of updates - once applied, an update persists until it is overwritten by a client.
  - Reliability - the system is guaranteed to function correctly as long as the majority of servers function correctly.

Zookeeper API

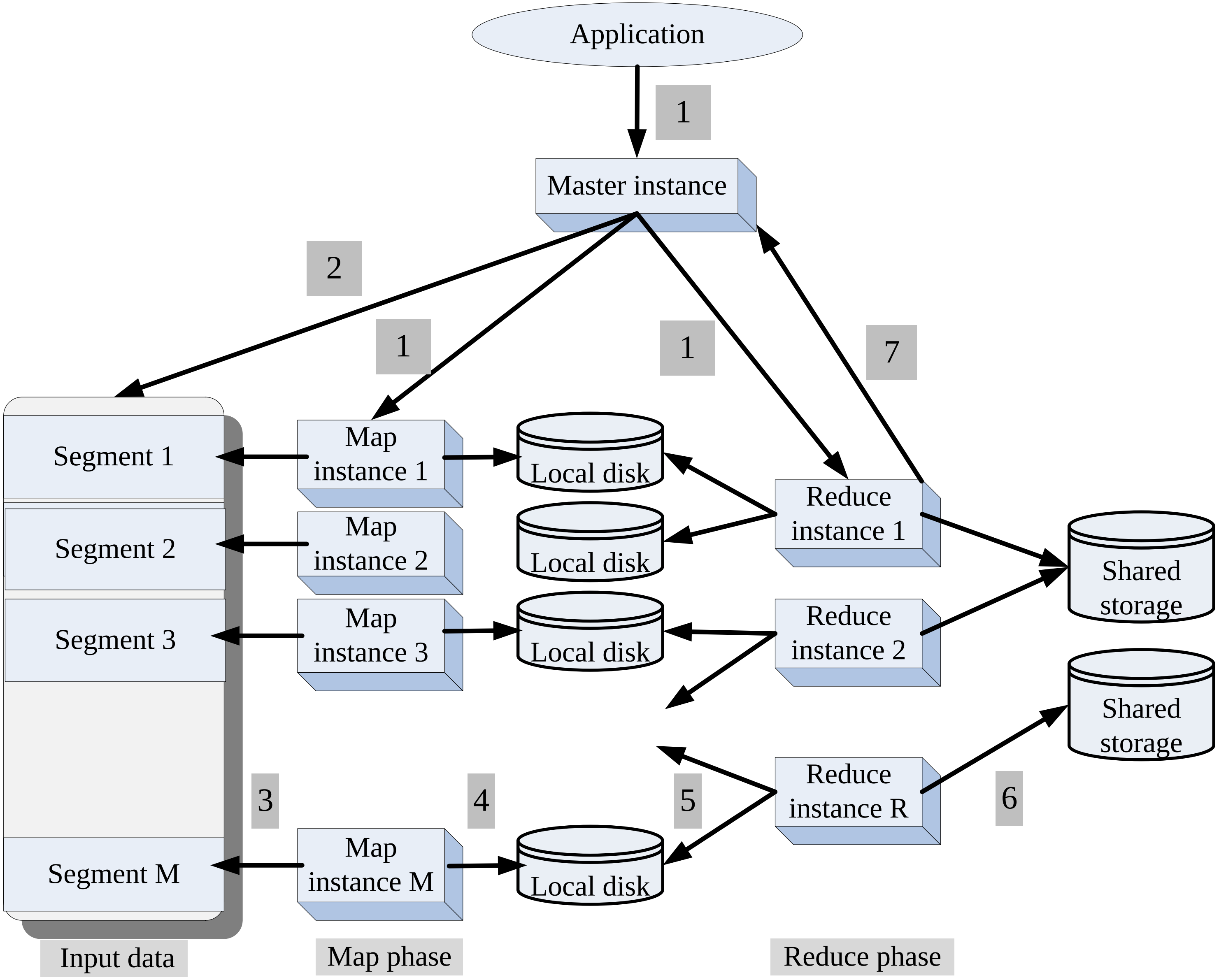
- The API is simple - consists of seven operations:
  - create - add a node at a given location on the tree;
  - delete - delete a node;
  - get data - read data from a node;
  - set data - write data to a node;
  - get children - retrieve a list of the children of the node
  - Synch - wait for the data to propagate.

Elasticity and load distribution

- Elasticity → ability to use as many servers as necessary to optimally respond to cost and timing constraints of application.
- How to divide the load
  - Transaction processing systems → a front-end distributes the incoming transactions to a number of back-end systems. As the workload increases new back-end systems are added to the pool.
  - For data-intensive batch applications two types of divisible workloads:
    - modularly divisible → the workload partitioning is defined apriori
    - arbitrarily divisible → the workload can be partitioned into an arbitrarily large number of smaller workloads of equal, or very close size.
- Many applications in physics, biology, and other areas of computational science and engineering obey the arbitrarily divisible load sharing model.

MapReduce philosophy

1. An application starts a master instance, M worker instances for the *Map phase,* and later R worker instances for the *Reduce phase*.
2. The master instance partitions the input data in M *segments*.
3. Each *map instance* reads its input data segment and processers the data.
4. The results of the processing are stored on the local disks of the servers where the map instances run.
5. When all map instances have finished processing their data the R reduce instances read the results of the first phase and merges the partial results.
6. The final results are written by the reduce instances to a shared storage server.
7. The master instance monitors the reduce instances and when all of them report task completion the application is terminated.


Application
1
Master instance
2
1
1
7
Segment 1
Segment 2
Segment 3
Segment M
Map instance 1
Map instance 2
Map instance 3
Map instance M
Local disk
Local disk
Local disk
Local disk
Reduce instance 1
Reduce instance 2
Reduce instance R
Shared storage
Shared storage
3
4
5
6
Input data
Map phase
Reduce phase

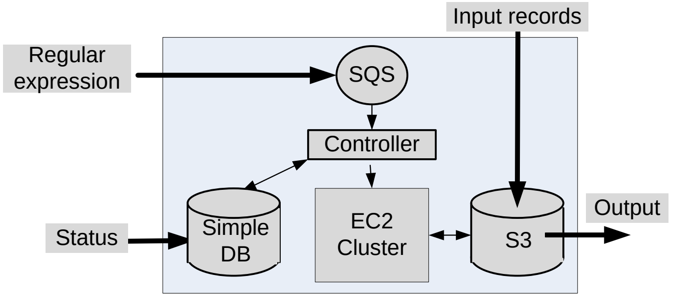
Case study: GrepTheWeb

- The application illustrates the means to
  - create an on-demand infrastructure
  - run it on a massively distributed system in a manner that allows it to run in parallel and scale up and down based on the number of users and the problem size
- GrepTheWeb
  - Performs a search of a very large set of records to identify records that satisfy a regular expression.
  - It is analogous to the Unix *grep* command.
  - The source is a collection of document URLs produced by the Alexa Web Search, a software system that crawls the web every night.
  - Uses message passing to trigger the activities of multiple controller threads which launch the application, initiate processing, shutdown the system, and create billing records.

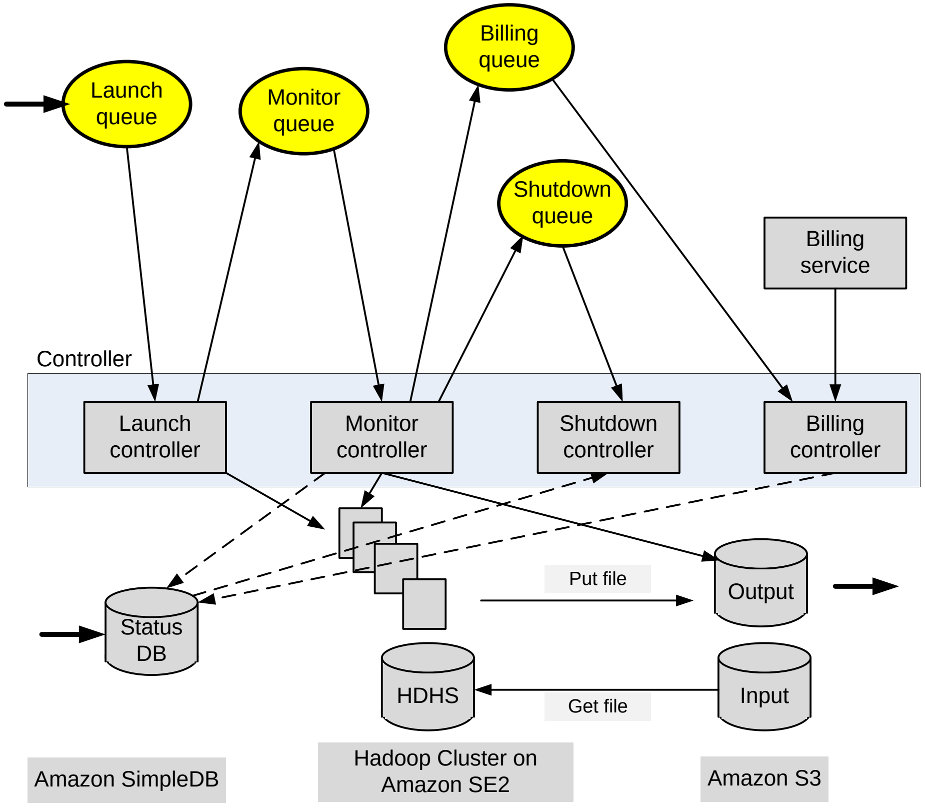
(a) The simplified workflow showing the inputs:

- the regular expression;
- the input records generated by the web crawler;
- the user commands to report the current status and to terminate the processing.

(b) The detailed workflow; the system is based on message passing between several queues; four controller threads periodically poll their associated input queues, retrieve messages, and carry out the required actions



(a)



(b)

Clouds for science and engineering

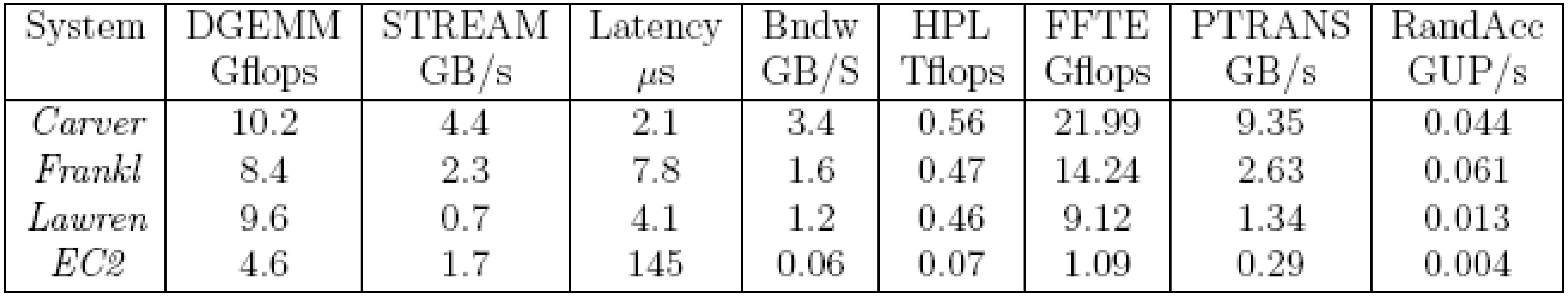
- The generic problems in virtually all areas of science are:
  - □ Collection of experimental data.
  - □ Management of a very large volumes of data.
  - □ Building and execution of models.
  - □ Integration of data and literature.
  - □ Documentation of the experiments.
  - □ Sharing the data with others; data preservation for a long periods of time.
- All these activities require “big” data storage and systems capable to deliver abundant computing cycles; computing clouds are able to provide such resources and support collaborative environments.

Online data discovery

- Phases of data discovery in large scientific data sets:
  - recognition of the information problem;
  - generation of search queries using one or more search engines;
  - evaluation of the search results;
  - evaluation of the web documents;
  - comparing information from different sources.
- Large scientific data sets:
  - biomedical and genomic data from the National Center for Biotechnology Information (NCBI)
  - astrophysics data from NASA
  - atmospheric data from the National Oceanic and Atmospheric Administration (NOAA) and the National Center for Atmospheric Research (NCAR).

High performance computing on a cloud

- Comparative benchmark of EC2 and three supercomputers at the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory. NERSC has some 3,000 researchers and involves 400 projects based on some 600 codes.
- Conclusion - communication intensive applications are affected by the increased latency and lower bandwidth of the cloud. The low latency and high bandwidth of the interconnection network of a supercomputer cannot be matched by a cloud.

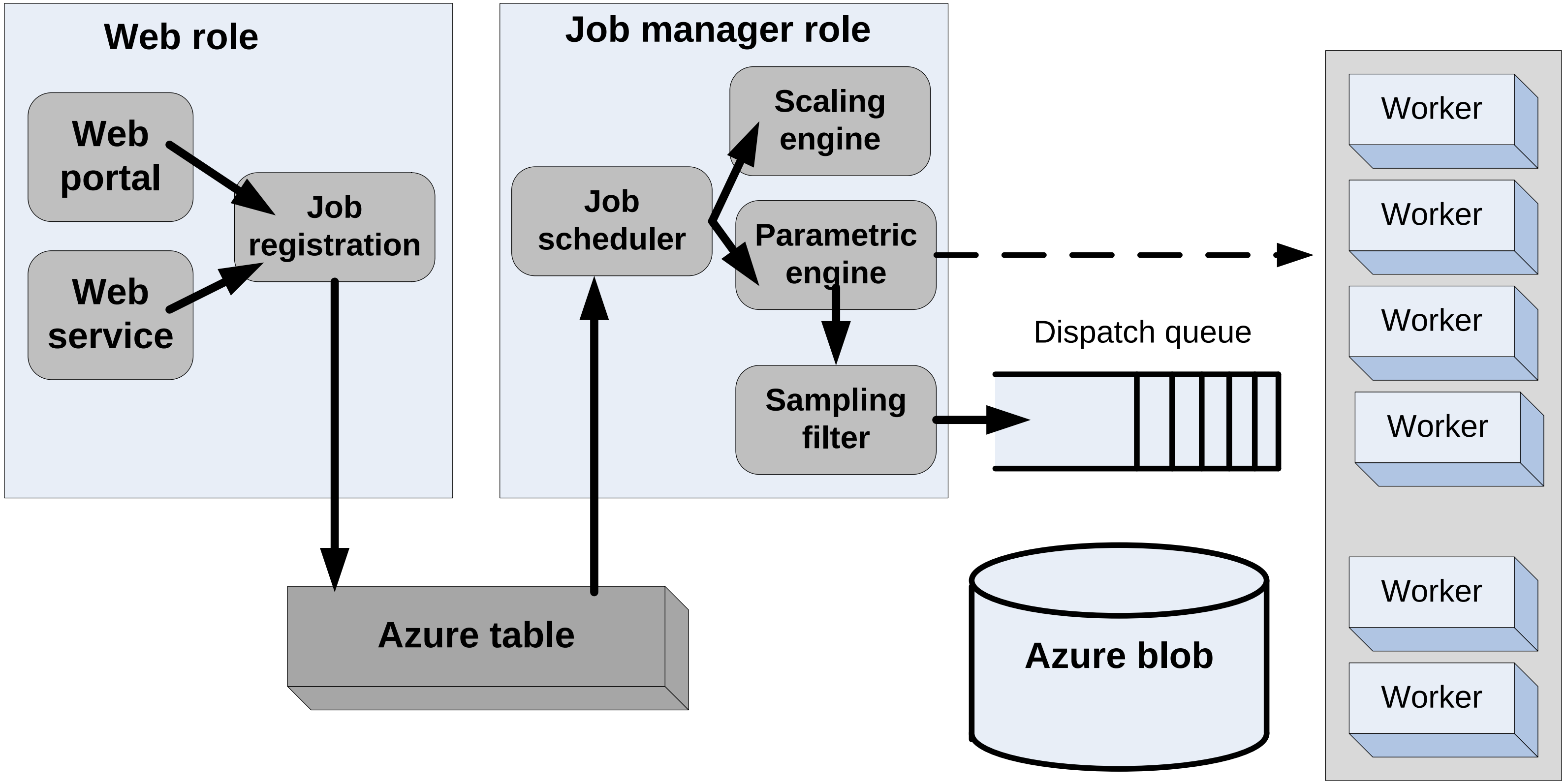


System	DGEMM Gflops	STREAM GB/s	Latency μs	Bndw GB/S	HPL Tflops	FFTE Gflops	PTRANS GB/s	RandAcc GUP/s
Carver	10.2	4.4	2.1	3.4	0.56	21.99	9.35	0.044
Frankl	8.4	2.3	7.8	1.6	0.47	14.24	2.63	0.061
Lawren	9.6	0.7	4.1	1.2	0.46	9.12	1.34	0.013
EC2	4.6	1.7	145	0.06	0.07	1.09	0.29	0.004

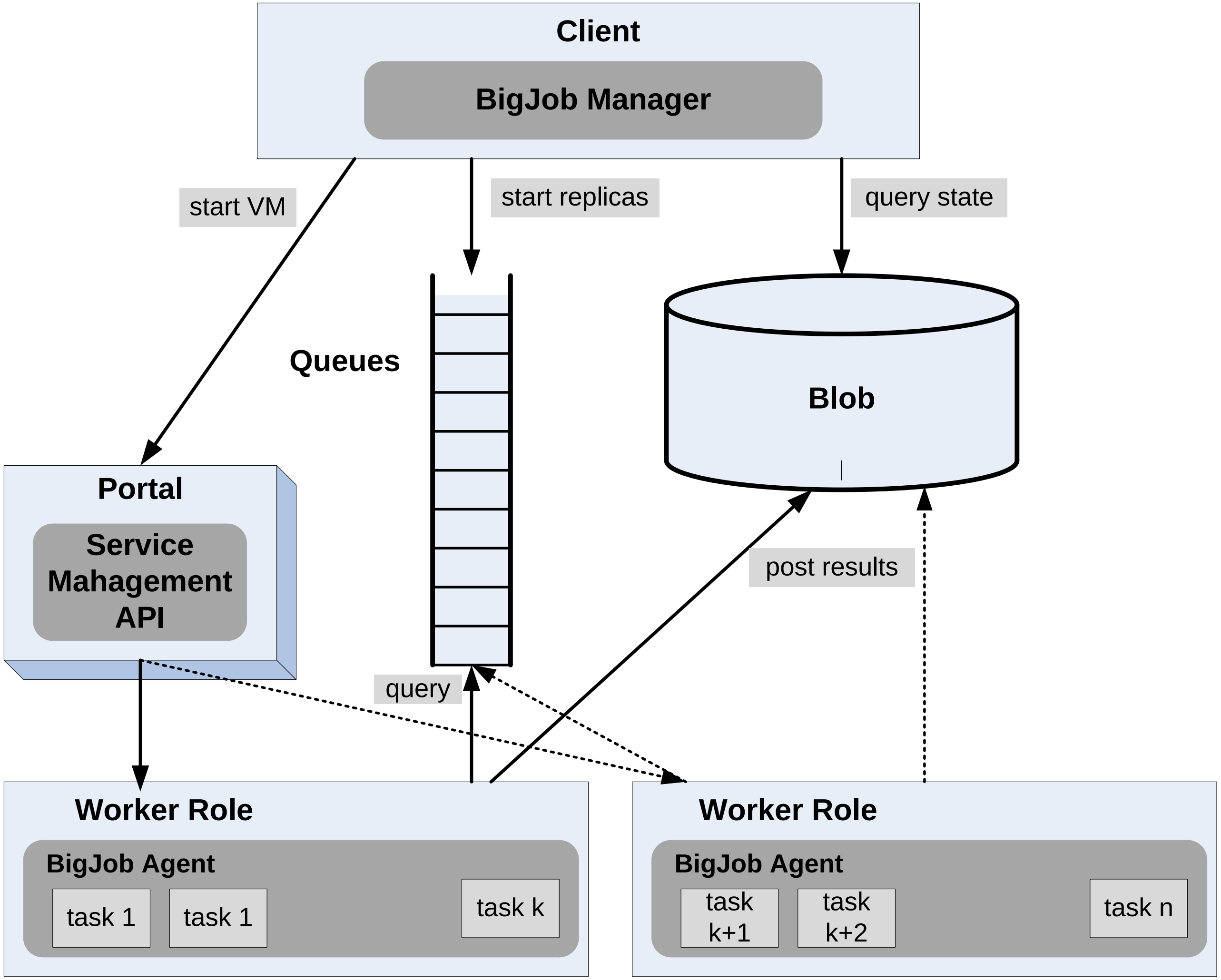
Legacy applications on the cloud

- Is it feasible to run legacy applications on a cloud?
- Cirrus - a general platform for executing legacy Windows applications on the cloud. A Cirrus job - a prologue, commands, and parameters. The prologue sets up the running environment; the commands are sequences of shell scripts including Azure-storage-related commands to transfer data between Azure blob storage and the instance.
- BLAST - a biology code which finds regions of local similarity between sequences; it compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches; used to infer functional and evolutionary relationships between sequences and identify members of gene families.
- AzureBLAST - a version of BLAST running on the Azure platform.

Cirrus


Web role
Web portal
Web service
Job registration
Job manager role
Scaling engine
Job scheduler
Parametric engine
Sampling filter
Dispatch queue
Azure table
Azure blob
Worker
Worker
Worker
Worker
Worker
Worker

Execution of loosely-coupled workloads using the Azure platform



Social computing and digital content

- Networks allowing researchers to share data and provide a virtual environment supporting remote execution of workflows are domaon specific:
  - MyExperiment for biology.
  - nanoHub for nanoscience.
- Volunteer computing - a large population of users donate resources such as CPU cycles and storage space for a specific project:
  - Mersenne Prime Search
  - SETI@Home,
  - Folding@home,
  - Storage@Home
  - PlanetLab
- Berkeley Open Infrastructure for Network Computing (BOINC) → middleware for a distributed infrastructure suitable for different applications.

4. Virtualization

- Virtual machine monitor
- Virtual machine
- Performance and security isolation
- Architectural support for virtualization
- x86 support for virtualization
- Full and paravirtualization
- Xen 1.0 and 2.0
- Performance comparison of virtual machine monitors
- The darker side of virtualization

Virtual machine monitor (VMM / hypervisor)

- Partitions the resources of computer system into one or more virtual machines (VMs). Allows several operating systems to run concurrently on a single hardware platform.
- A VMM allows
  - Multiple services to share the same platform.
  - Live migration- the movement of a server from one platform to another.
  - System modification while maintaining backward compatibility with the original system.
  - Enforces isolation among the systems, thus security.

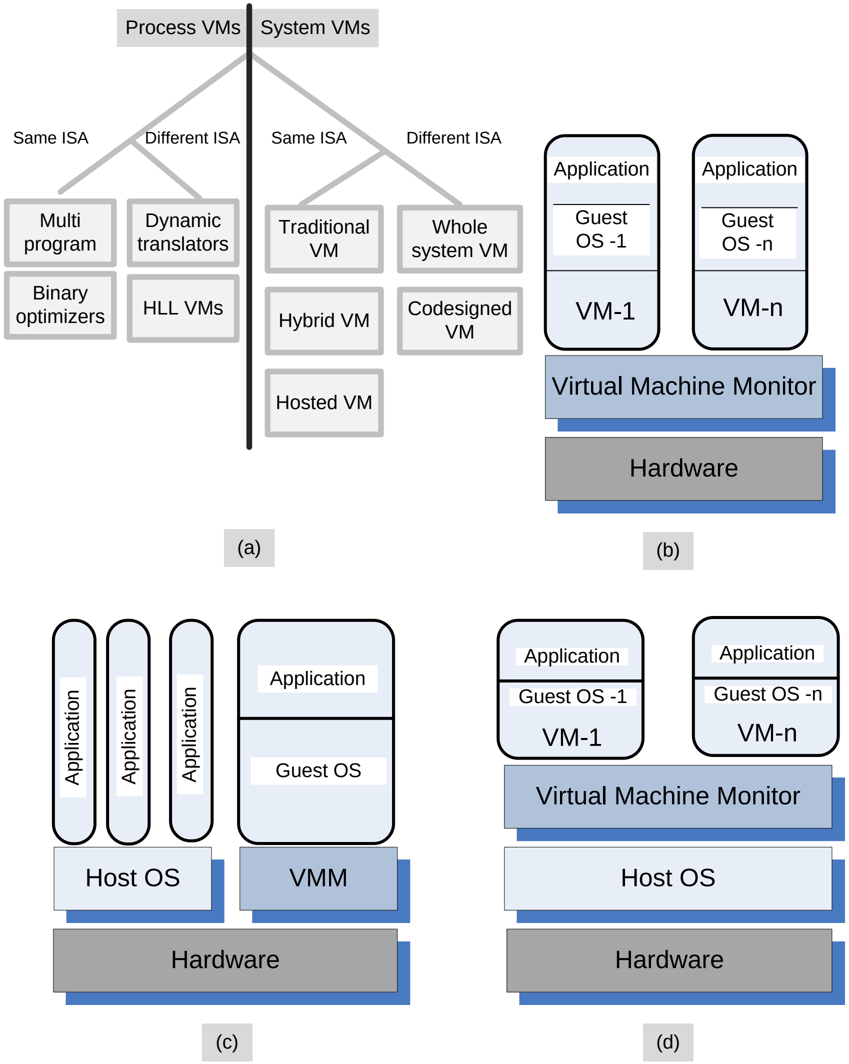
VMM virtualizes the CPU and the memory

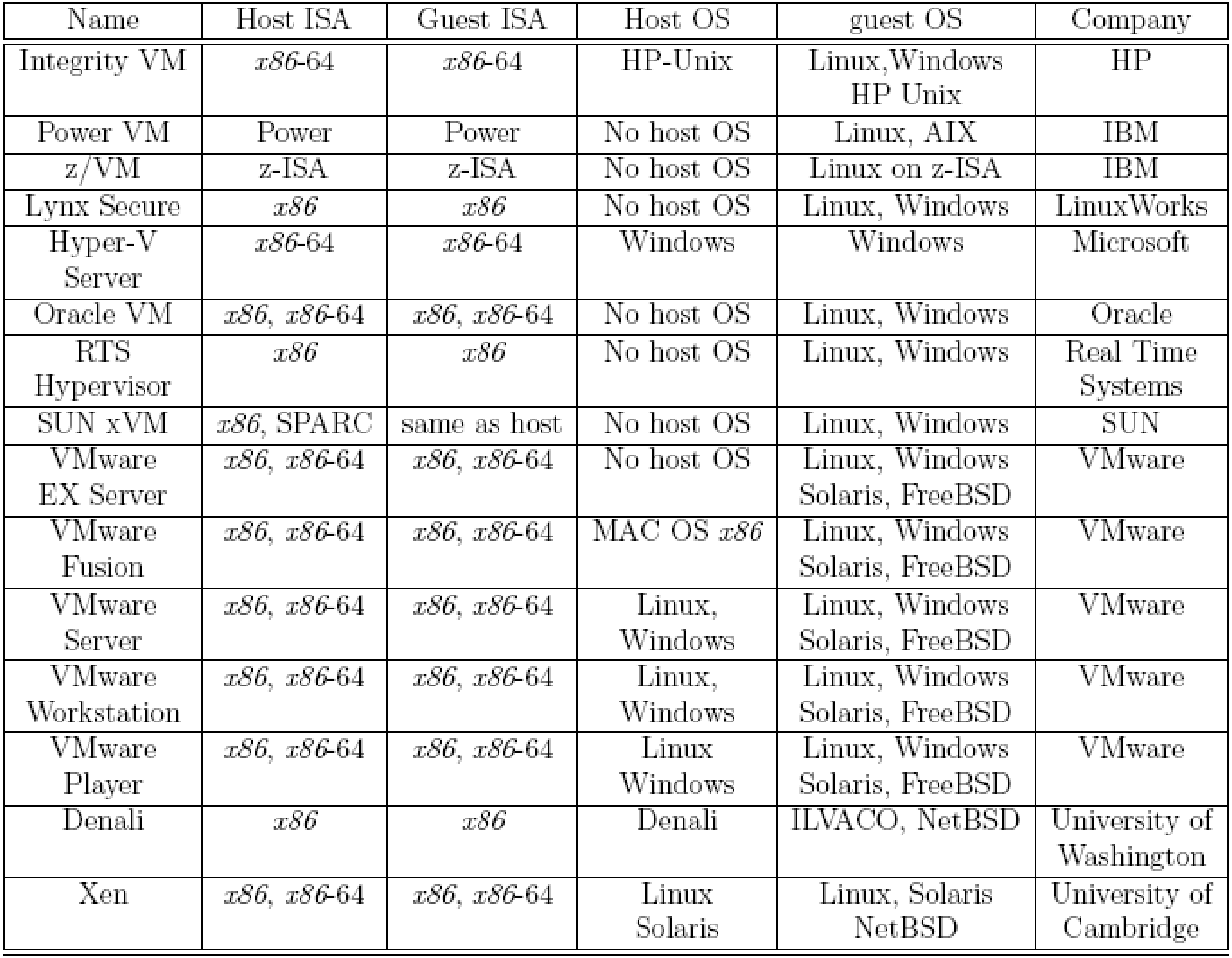
- A VMM
  - □ Traps the privileged instructions executed by a guest OS and enforces the correctness and safety of the operation.
  - □ Traps interrupts and dispatches them to the individual guest operating systems.
  - □ Controls the virtual memory management.
  - □ Maintains a shadow page table for each guest OS and replicates any modification made by the guest OS in its own shadow page table; this shadow page table points to the actual page frame and it is used by the Memory Management Unit (MMU) for dynamic address translation.
  - □ Monitors the system performance and takes corrective actions to avoid performance degradation. For example, the VMM may swap out a Virtual Machine to avoid thrashing.

Virtual machines (VMs)

- VM - isolated environment that appears to be a whole computer, but actually only has access to a portion of the computer resources.
- Process VM - a virtual platform created for an individual process and destroyed once the process terminates.
- System VM - supports an operating system together with many user processes.
- Traditional VM - supports multiple virtual machines and runs directly on the hardware.
- Hybrid VM - shares the hardware with a host operating system and supports multiple virtual machines.
- Hosted VM - runs under a host operating system.

Traditional, hybrid, and hosted VMs





Name	Host ISA	Guest ISA	Host OS	guest OS	Company
Integrity VM	*x86-64*	*x86-64*	HP-Unix	Linux,Windows HP Unix	HP
Power VM	Power	Power	No host OS	Linux, AIX	IBM
z/VM	z-ISA	z-ISA	No host OS	Linux on z-ISA	IBM
Lynx Secure	*x86*	*x86*	No host OS	Linux, Windows	LinuxWorks
Hyper-V Server	*x86-64*	*x86-64*	Windows	Windows	Microsoft
Oracle VM	*x86*, *x86-64*	*x86*, *x86-64*	No host OS	Linux, Windows	Oracle
RTS Hypervisor	*x86*	*x86*	No host OS	Linux, Windows	Real Time Systems
SUN xVM	*x86*, SPARC	same as host	No host OS	Linux, Windows	SUN
VMware EX Server	*x86*, *x86-64*	*x86*, *x86-64*	No host OS	Linux, Windows Solaris, FreeBSD	VMware
VMware Fusion	*x86*, *x86-64*	*x86*, *x86-64*	MAC OS *x86*	Linux, Windows Solaris, FreeBSD	VMware
VMware Server	*x86*, *x86-64*	*x86*, *x86-64*	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Workstation	*x86*, *x86-64*	*x86*, *x86-64*	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Player	*x86*, *x86-64*	*x86*, *x86-64*	Linux Windows	Linux, Windows Solaris, FreeBSD	VMware
Denali	*x86*	*x86*	Denali	ILVACO, NetBSD	University of Washington
Xen	*x86*, *x86-64*	*x86*, *x86-64*	Linux Solaris	Linux, Solaris NetBSD	University of Cambridge

Performance and security isolation

- The run-time behavior of an application is affected by other applications running concurrently on the same platform and competing for CPU cycles, cache, main memory, disk and network access. Thus, it is difficult to predict the completion time!
- Performance isolation - a critical condition for QoS guarantees in shared computing environments.
- A VMM is a much simpler and better specified system than a traditional operating system. Example - Xen has approximately 60,000 lines of code; Denali has only about half, 30,000.
- The security vulnerability of VMMs is considerably reduced as the systems expose a much smaller number of privileged functions.

Computer architecture and virtualization

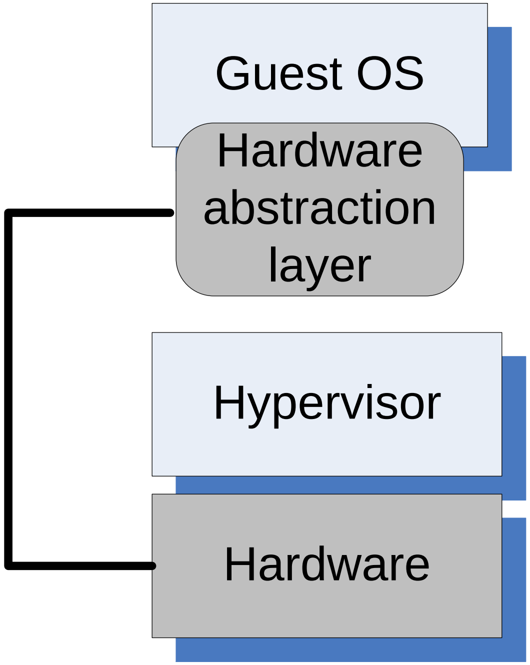
- Conditions for efficient virtualization
  - A program running under the VMM should exhibit a behavior essentially identical to that demonstrated when running on an equivalent machine directly.
  - The VMM should be in complete control of the virtualized resources.
  - A statistically significant fraction of machine instructions must be executed without the intervention of the VMM.
- Two classes of machine instructions:
  - Sensitive - require special precautions at execution time:
    - Control sensitive - instructions that attempt to change either the memory allocation or the privileged mode.
    - Mode sensitive - instructions whose behavior is different in the privileged mode.
  - Innocuous - not sensitive.

Full virtualization and paravirtualization

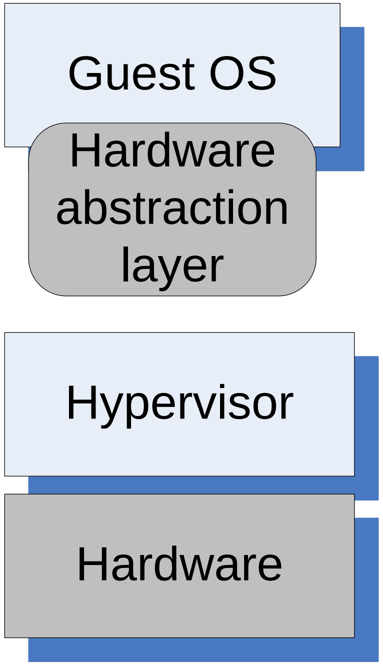
- Full virtualization – a guest OS can run unchanged under the VMM as if it was running directly on the hardware platform.
  - Requires a virtualizable architecture
  - Examples: VMware
- Paravirtualization - a guest operating system is modified to use only instructions that can be virtualized. Reasons for paravirtualization:
  - Some aspects of the hardware cannot be virtualized.
  - Improved performance.
  - Present a simpler interface

  Examples: Xen, Denaly

Full virtualization and paravirtualization



(a) Full virtualization



(b) Paravirtualization

Virtualization of x86 architecture

- Ring de-privileging - a VMMs forces the operating system and the applications, to run at a privilege level greater than 0.
- Ring aliasing - a guest OS is forced to run at a privilege level other than that it was originally designed for.
- Address space compression - a VMM uses parts of the guest address space to store several system data structures.
- Non-faulting access to privileged state - several store instructions can only be executed at privileged level 0 because they operate on data structures that control the CPU operation. They fail silently when executed at a privilege level other than 0.
- Guest system calls which cause transitions to/from privilege level 0 must be emulated by the VMM.
- Interrupt virtualization - in response to a physical interrupt the VMM generates a ``virtual interrupt'' and delivers it later to the target guest OS which can mask interrupts.

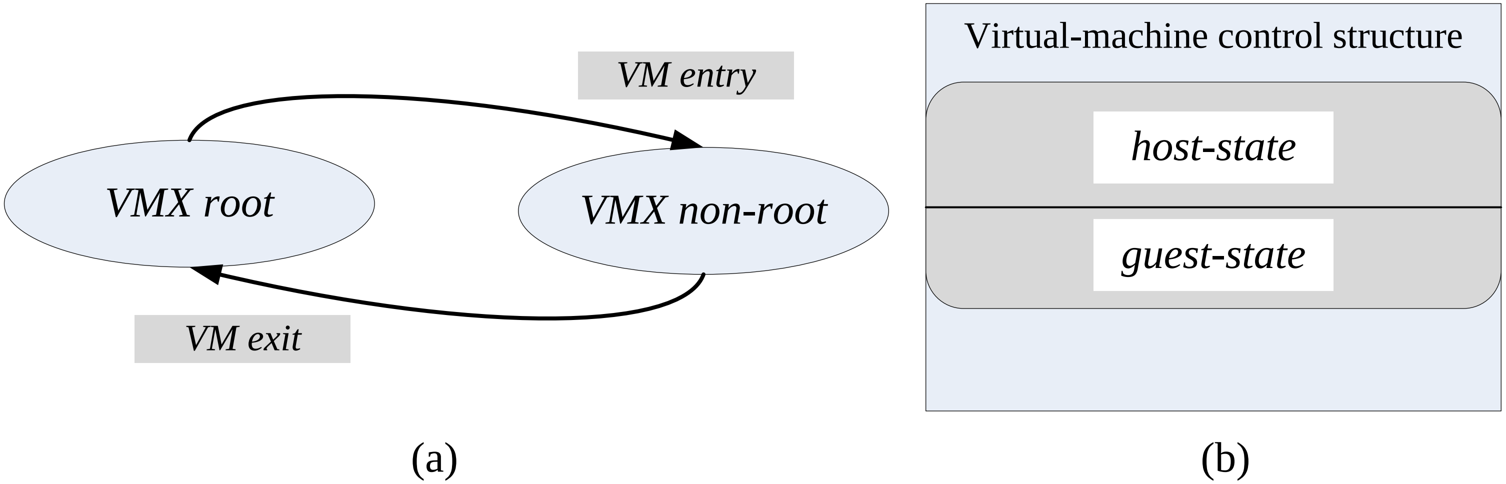
Virtualization of x86 architecture (cont'd)

- Access to hidden state - elements of the system state, e.g., descriptor caches for segment registers, are hidden; there is no mechanism for saving and restoring the hidden components when there is a context switch from one VM to another.
- Ring compression - paging and segmentation protect VMM code from being overwritten by guest OS and applications. Systems running in 64-bit mode can only use paging, but paging does not distinguish between privilege levels 0, 1, and 2, thus the guest OS must run at privilege level 3, the so called (0/3/3) mode. Privilege levels 1 and 2 cannot be used thus, the name ring compression.
- The task-priority register is frequently used by a guest OS; the VMM must protect the access to this register and trap all attempts to access it. This can cause a significant performance degradation.

VT-x a major architectural enhancement

- Supports two modes of operations:
  - VMX root - for VMM operations
  - VMX non-root - support a VM.
- The Virtual Machine Control Structure including host-state and guest-state areas.
  - VM entry - the processor state is loaded from the guest-state of the VM scheduled to run; then the control is transferred from VMM to the VM.
  - VM exit - saves the processor state in the guest-state area of the running VM; then it loads the processor state from the host-state area, finally transfers control to the VMM.

VT- x


VM entry
VMX root
VMX non-root
VM exit
(a)
Virtual-machine control structure
host-state
guest-state
(b)

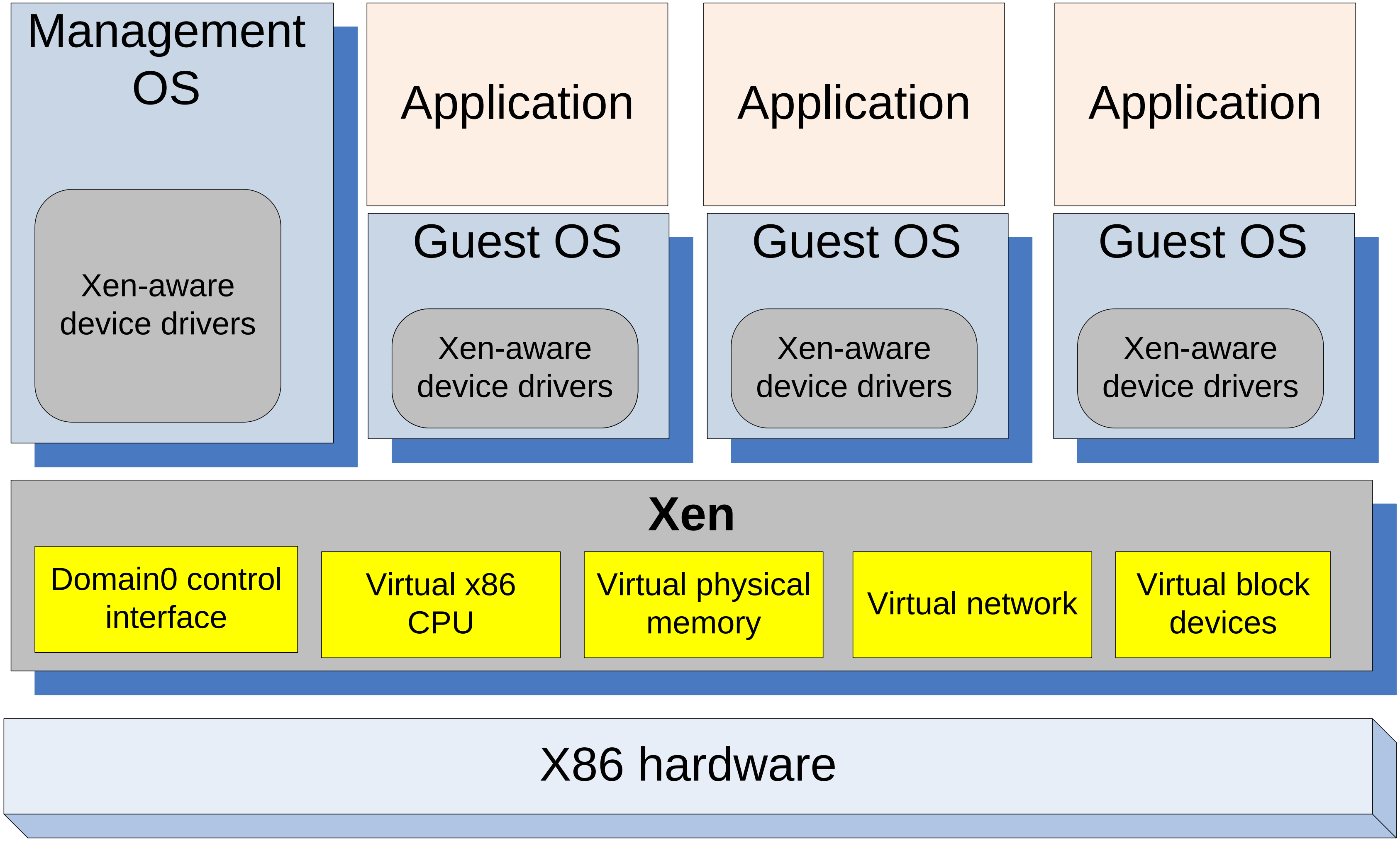
VT-d a new virtualization architectures

- I/O MMU virtualization gives VMs direct access to peripheral devices.
- VT-d supports:
  - DMA address remapping, address translation for device DMA transfers.
  - Interrupt remapping, isolation of device interrupts and VM routing.
  - I/O device assignment, the devices can be assigned by an administrator to a VM in any configurations.
  - Reliability features, it reports and records DMA and interrupt errors that my otherwise corrupt memory and impact VM isolation.

Xen - a VMM based on paravirtualization

- The goal of the Cambridge group - design a VMM capable of scaling to about 100 VMs running standard applications and services without any modifications to the Application Binary Interface (ABI).
- Linux, Minix, NetBSD, FreeBSD, NetWare, and OZONE can operate as paravirtualized Xen guest OS running on x86, x86-64, Itanium, and ARM architectures.
- Xen domain - ensemble of address spaces hosting a guest OS and applications running under the guest OS. Runs on a virtual CPU.
    - Dom0 - dedicated to execution of Xen control functions and privileged instructions
    - DomU - a user domain
- Applications make system calls using hypercalls processed by Xen; privileged instructions issued by a guest OS are paravirtualized and must be validated by Xen.

Xen


Management OS
Xen-aware device drivers
Application
Guest OS
Xen-aware device drivers
Application
Guest OS
Xen-aware device drivers
Application
Guest OS
Xen-aware device drivers
Xen
Domain0 control interface
Virtual x86 CPU
Virtual physical memory
Virtual network
Virtual block devices
X86 hardware

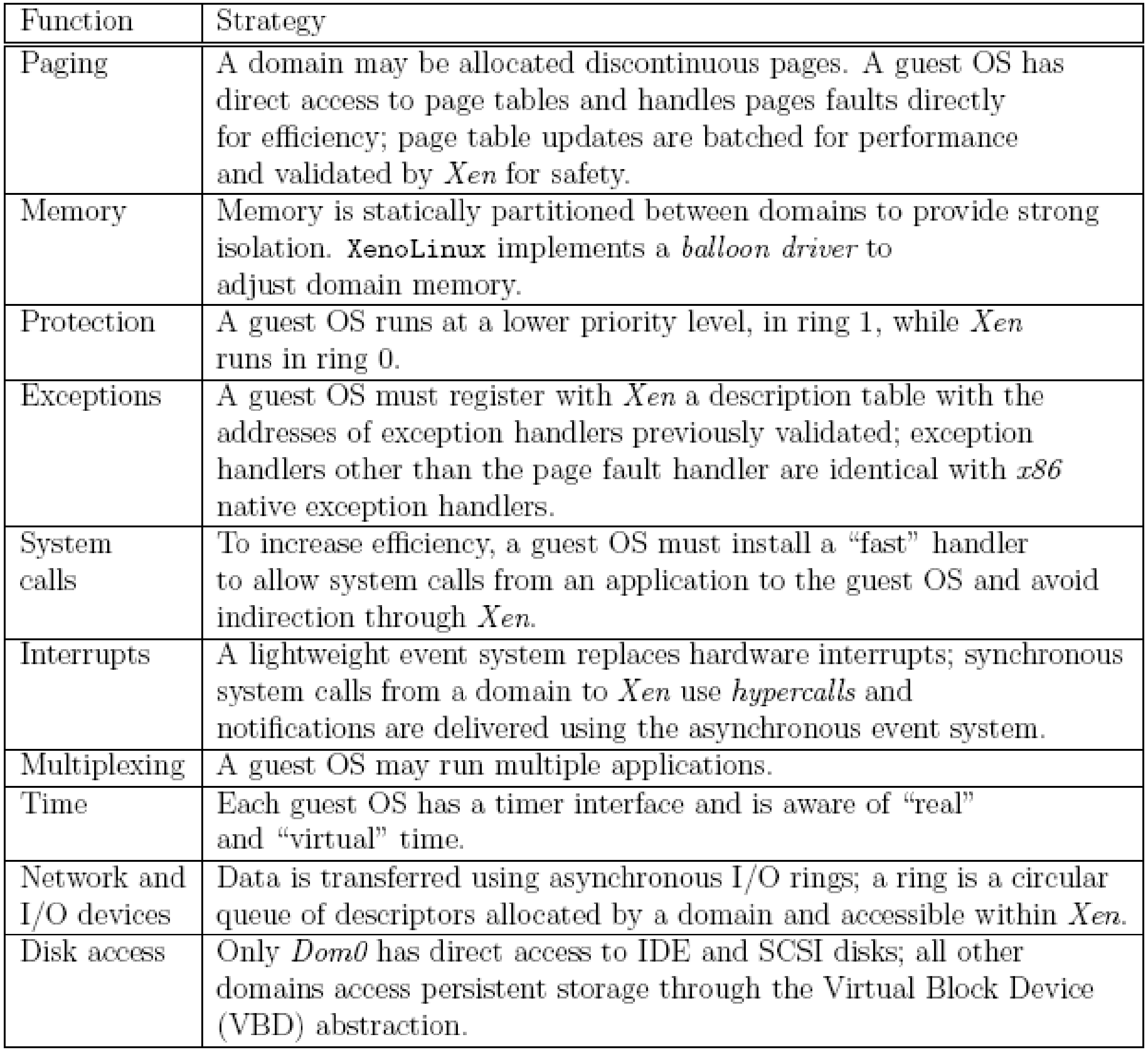
Xen implementation on x86 architecture

- Xen runs at privilege Level 0, the guest OS at Level 1, and applications at Level 3.
- The x86 architecture does not support either the tagging of TLB entries or the software management of the TLB; thus, address space switching, when the VMM activates a different OS, requires a complete TLB flush; this has a negative impact on the performance.
- Solution - load Xen in a 64 MB segment at the top of each address space and to delegate the management of hardware page tables to the guest OS with minimal intervention from Xen. This region is not accessible, or re-mappable by the guest OS.
- Xen schedules individual domains using the Borrowed Virtual Time (BVT) scheduling algorithm.
- A guest OS must register with Xen a description table with the addresses of exception handlers for validation.

Dom0 components

- XenStore – a Dom0 process.
  - Supports a system-wide registry and naming service.
  - Implemented as a hierarchical key-value storage.
  - A watch function of informs listeners of changes of the key in storage they have subscribed to.
  - Communicates with guest VMs via shared memory using Dom0 privileges.
- Toolstack - responsible for creating, destroying, and managing the resources and privileges of VMs.
  - To create a new VM a user provides a configuration file describing memory and CPU allocations and device configurations.
  - Toolstack parses this file and writes this information in XenStore.
  - Takes advantage of Dom0 privileges to map guest memory, to load a kernel and virtual BIOS and to set up initial communication channels with XenStore and with the virtual console when a new VM is created.

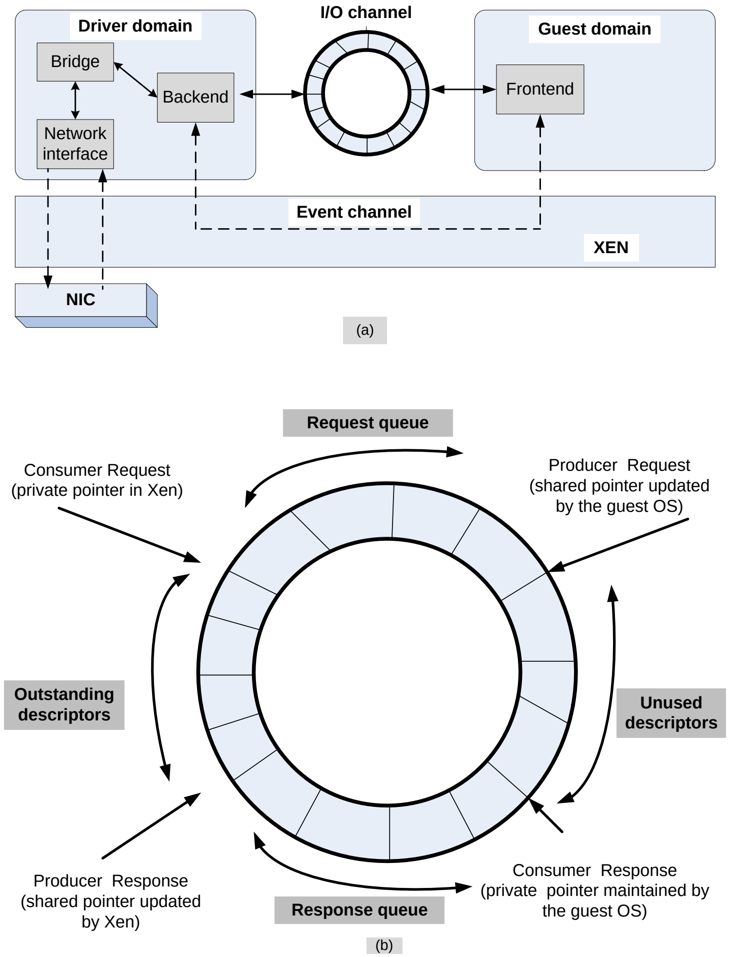
Strategies for virtual memory management, CPU multiplexing, and I/O devices



Function	Strategy
Paging	A domain may be allocated discontinuous pages. A guest OS has direct access to page tables and handles pages faults directly for efficiency; page table updates are batched for performance and validated by *Xen* for safety.
Memory	Memory is statically partitioned between domains to provide strong isolation. `XenoLinux` implements a *balloon driver* to adjust domain memory.
Protection	A guest OS runs at a lower priority level, in ring 1, while *Xen* runs in ring 0.
Exceptions	A guest OS must register with *Xen* a description table with the addresses of exception handlers previously validated; exception handlers other than the page fault handler are identical with *x86* native exception handlers.
System calls	To increase efficiency, a guest OS must install a "fast" handler to allow system calls from an application to the guest OS and avoid indirection through *Xen*.
Interrupts	A lightweight event system replaces hardware interrupts; synchronous system calls from a domain to *Xen* use *hypercalls* and notifications are delivered using the asynchronous event system.
Multiplexing	A guest OS may run multiple applications.
Time	Each guest OS has a timer interface and is aware of "real" and "virtual" time.
Network and I/O devices	Data is transferred using asynchronous I/O rings; a ring is a circular queue of descriptors allocated by a domain and accessible within *Xen*.
Disk access	Only *Dom0* has direct access to IDE and SCSI disks; all other domains access persistent storage through the Virtual Block Device (VBD) abstraction.

Xen abstractions for networking and I/O

- Each domain has one or more Virtual Network Interfaces (VIFs) which support the functionality of a network interface card. A VIF is attached to a Virtual Firewall-Router (VFR).
- Split drivers have a front-end of in the DomU and the back-end in Dom0; the two communicate via a ring in shared memory.
- Ring - a circular queue of descriptors allocated by a domain and accessible within Xen. Descriptors do not contain data , the data buffers are allocated off-band by the guest OS.
- Two rings of buffer descriptors, one for packet sending and one for packet receiving, are supported.
- To transmit a packet:
  - a guest OS enqueues a buffer descriptor to the send ring,
  - then Xen copies the descriptor and checks safety,
  - copies only the packet header, not the payload, and
  - executes the matching rules.


I/O channel
Driver domain
Bridge
Backend
Network interface
Guest domain
Frontend
Event channel
XEN
NIC
(a)
Request queue
Consumer Request (private pointer in Xen)
Producer Request (shared pointer updated by the guest OS)
Outstanding descriptors
Unused descriptors
Producer Response (shared pointer updated by Xen)
Response queue
Consumer Response (private pointer maintained by the guest OS)
(b)

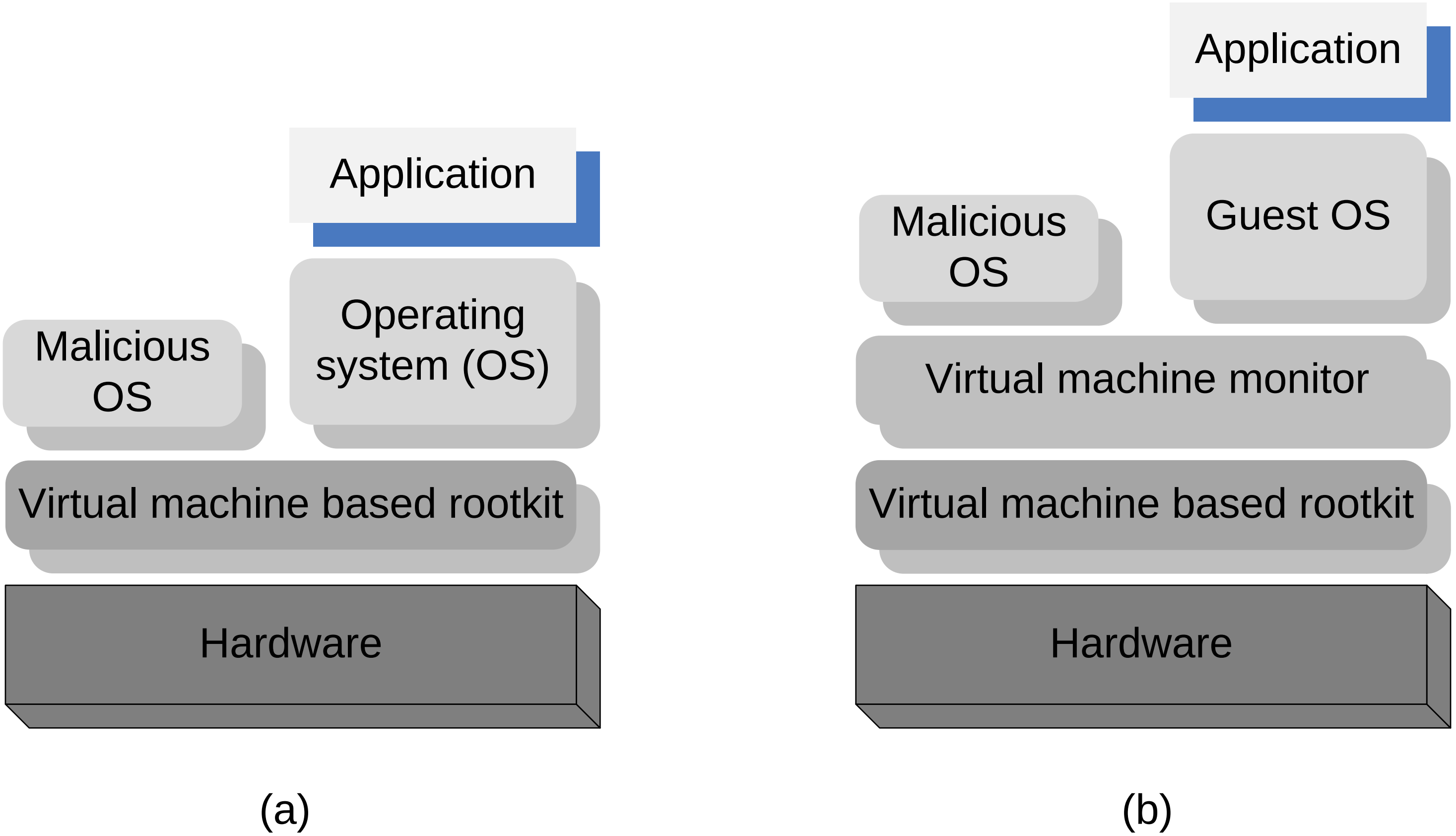
Xen zero-copy semantics for data transfer using I/O rings. (a) The communication between a guest domain and the driver domain over an I/O and an event channel; NIC is the Network Interface Controller. (b) the circular ring of buffers.

Xen 2.0

- Optimization of:
  - Virtual interface - takes advantage of the capabilities of some physical NICs such as checksum offload.
  - I/O channel - rather than copying a data buffer holding a packet, each packet is allocated in a new page and then the physical page containing the packet is re-mapped into the target domain.
  - Virtual memory - takes advantage of the superpage and global page mapping hardware on Pentium and Pentium Pro processors. A superpage entry covers 1,024 pages of physical memory and the address translation mechanism maps a set of contiguous pages to a set of contiguous physical pages. This helps reduce the number of TLB misses.

The darker side of virtualization

- In a layered structure a defense mechanism at some layer can be disabled by malware running at a layer below it.
- It is feasible to insert a *rogue VMM,* a Virtual-Machine Based Rootkit (VMBR) between the physical hardware and an operating system.
- Rootkit - malware with a privileged access to a system.
- The VMBR can enable a separate malicious OS to run surreptitiously and make this malicious OS invisible to the guest OS and to the application running under it.
- Under the protection of the VMBR the malicious OS could:
  - observe the data, the events, or the state of the target system;
  - run services such as spam relays or distributed denial-of-service attacks;
  - interfere with the application.



The insertion of a Virtual-Machine Based Rootkit (VMBR) as the lowest layer of the software stack running on the physical hardware; (a) below an operating system; (b) below a legitimate virtual machine monitor. The VMBR enables a malicious OS to run surreptitiously and makes it invisible to the genuine or the guest OS and to the application.

5. Cloud resource management

- Policies and mechanisms
- Tradeoffs
- Resource bundling
- Combinatorial auctions

Motivation

- Cloud resource management
  - Requires complex policies and decisions for multi-objective optimization.
  - It is challenging - the complexity of the system makes it impossible to have accurate global state information and because of the
  - Affected by unpredictable interactions with the environment, e.g., system failures, attacks
  - Cloud service providers are faced with large fluctuating loads which challenge the claim of cloud elasticity.
- The strategies for resource management for IaaS, PaaS, and SaaS are different.

Cloud resource management (CRM) policies

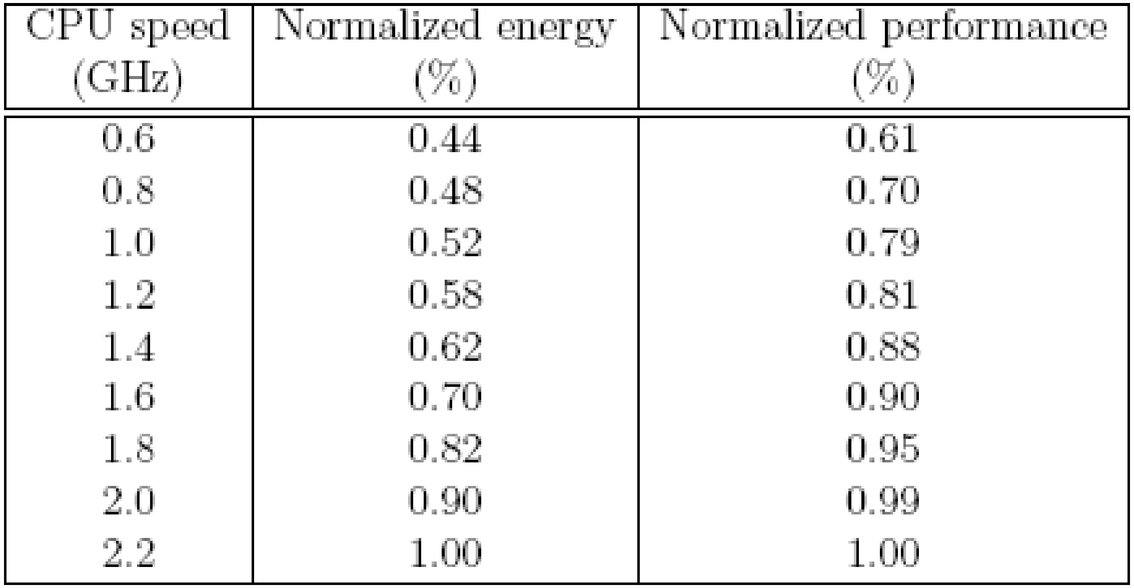
1. Admission control → prevent the system from accepting workload in violation of high-level system policies.
2. Capacity allocation → allocate resources for individual activations of a service
3. Load balancing → distributing the workload evenly among the servers
4. Energy optimization → minimization of energy consumption
5. Quality of service (QoS) guarantees → ability to satisfy timing or other conditions specified by a Service Level Agreement.

Mechanisms for the implementation of resource management policies

- Control theory → uses the feedback to guarantee system stability and predict transient behavior.
- Machine learning → does not need a performance model of the system.
- Utility-based → require a performance model and a mechanism to correlate user-level performance with cost.
- Market-oriented/economic → do not require a model of the system, e.g., combinatorial auctions for bundles of resources

Tradeoffs

- To reduce cost and save energy we may need to concentrate the load on fewer servers rather than balance the load among them.
- We may also need to operate at a lower clock rate; the performance decreases at a lower rate than does the energy.



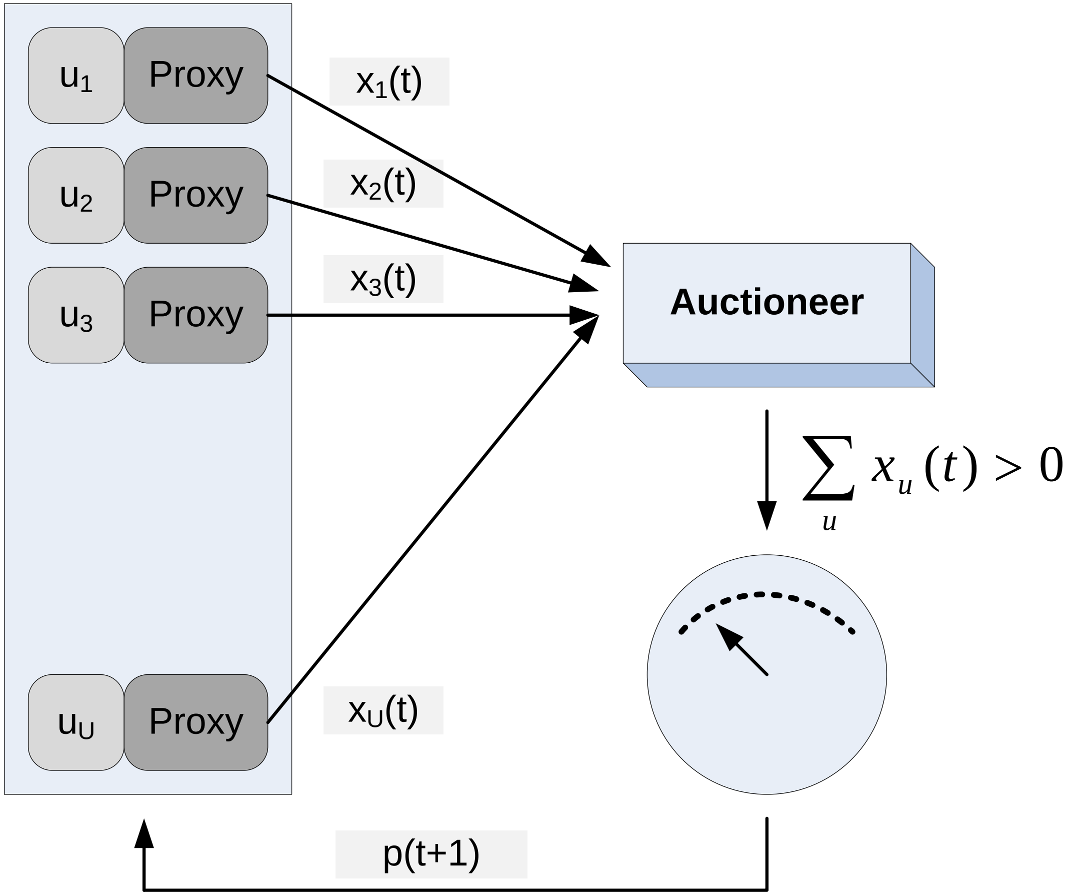
CPU speed (GHz)	Normalized energy (%)	Normalized performance (%)
0.6	0.44	0.61
0.8	0.48	0.70
1.0	0.52	0.79
1.2	0.58	0.81
1.4	0.62	0.88
1.6	0.70	0.90
1.8	0.82	0.95
2.0	0.90	0.99
2.2	1.00	1.00

Resource bundling

- Resources in a cloud are allocated in bundles.
- Users get maximum benefit from a specific combination of resources: CPU cycles, main memory, disk space, network bandwidth, and so on.
- Resource bundling complicates traditional resource allocation models and has generated an interest in economic models and, in particular, in auction algorithms.
- The bidding process aims to optimize an objective function f(x,p).
- In the context of cloud computing, an auction is the allocation of resources to the highest bidder.

Combinatorial auctions for cloud resources

- Users provide bids for desirable bundles and the price they are willing to pay
- Prices and allocation are set as a result of an auction.
- Ascending Clock Auction, (ASCA) → the current price for each resource is represented by a "clock" seen by all participants at the auction.
- The algorithm involves user bidding in multiple rounds; to address this problem the user proxies automatically adjust their demands on behalf of the actual bidders.



The schematics of the ASCA algorithm; to allow for a single round auction users are represented by proxies which place the bids $x_u(t)$. The auctioneer determines if there is an excess demand and, in that case, it raises the price of resources for which the demand exceeds the supply and requests new bids.

Pricing and allocation algorithms

- A pricing and allocation algorithm partitions the set of users in two disjoint sets, winners and losers.
- Desirable properties of a pricing algorithm:
  - Be computationally tractable; traditional combinatorial auction algorithms e.g., Vickey-Clarke-Groves (VLG) are not computationally tractable.
  - Scale well - given the scale of the system and the number of requests for service, scalability is a necessary condition.
  - Be objective - partitioning in winners and losers should only be based on the price of a user's bid; if the price exceeds the threshold then the user is a winner, otherwise the user is a loser.
  - Be fair - make sure that the prices are uniform, all winners within a given resource pool pay the same price.
  - Indicate clearly at the end of the auction the unit prices for each resource pool.
  - Indicate clearly to all participants the relationship between the supply and the demand in the system.

6. Cloud security

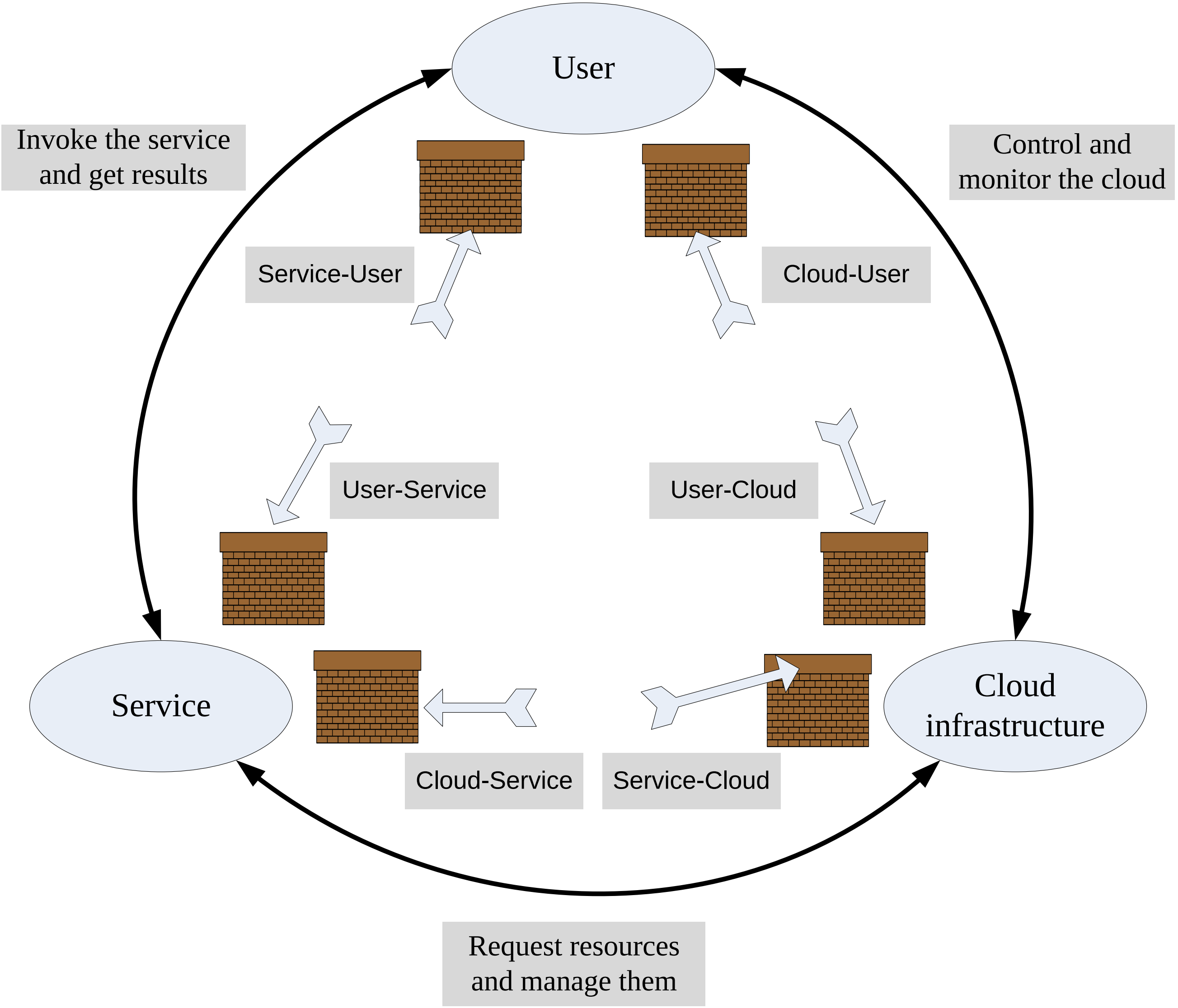
- Cloud security risks
- Operating systems security.
- Virtual machine security.
- Security of virtualization
- Security risks posed by shared images
- Security risks posed by a management OS
- XOAR- breaking the monolithic design of TCB

Cloud security risks

- Traditional threats → impact amplified due to the vast amount of cloud resources and the large user population that can be affected. The fuzzy bounds of responsibility between the providers of cloud services and users and the difficulties to accurately identify the cause.
- New threats → cloud servers host multiple VMs; multiple applications may run under each VM. Multi-tenancy and VMM vulnerabilities open new attack channels for malicious users. Identifying the path followed by an attacker more difficult in a cloud environment.
- Authentication and authorization → the procedures in place for one individual does not extend to an enterprise.
- Third-party control → generates a spectrum of concerns caused by the lack of transparency and limited user control.
- Availability of cloud services → system failures, power outages, and other catastrophic events could shutdown services for extended periods of time.

Attacks in a cloud computing environment

- Three actors involved; six types of attacks possible.
  - The user can be attacked by:
    - Service → SSL certificate spoofing, attacks on browser caches, or phishing attacks.
    - The cloud infrastructure → attacks that either originates at the cloud or spoofs to originate from the cloud infrastructure.
  - The service can be attached by:
    - A user→ buffer overflow, SQL injection, and privilege escalation are the common types of attacks.
    - The cloud infrastructure → the most serious line of attack. Limiting access to resources, privilege-related attacks, data distortion, injecting additional operations.
  - The cloud infrastructure can be attached by:
    - A user → targets the cloud control system.
    - A service → requesting an excessive amount of resources and causing the exhaustion of the resources.



Surfaces of attacks in a cloud computing environment.

Top threats to cloud computing

- Identified by a 2010n Cloud Security Alliance (CSA) report:
  - The abusive use of the cloud -the ability to conduct nefarious activities from the cloud
  - APIs that are not fully secure - may not protect the users during a range of activities starting with authentication and access control to monitoring and control of the application during runtime.
  - Malicious insiders - cloud service providers do not disclose their hiring standards and policies so this can be a serious threat
  - Shared technology.
  - Account hijacking.
  - Data loss or leakage – if the only copy of the data is stored on the cloud, then sensitive data is permanently lost when cloud data replication fails followed by a storage media failure
  - Unknown risk profile - exposure to the ignorance or underestimation of the risks of cloud computing

Auditability of cloud activities

- The lack of transparency makes auditability a very difficult proposition for cloud computing.
- Auditing guidelines elaborated by the National Institute of Standards (NIST) are mandatory for US Government agencies:
  - the Federal Information Processing Standard (FIPS)
  - the Federal Information Security Management Act (FISMA)

Security - the top concern for cloud users

- The unauthorized access to confidential information and the data theft top the list of user concerns.
  - Data is more vulnerable in storage, as it is kept in storage for extended periods of time.
  - Threats during processing cannot be ignored; such threats can originate from flaws in the VMM, rogue VMs, or a VMBR.
- There is the risk of unauthorized access and data theft posed by rogue employees of a Cloud Service Provider (CSP).
- Lack of standardization is also a major concern.
- Users are concerned about the legal framework for enforcing cloud computing security.
- Multi-tenancy is the root cause of many user concerns. Nevertheless, multi-tenancy enables a higher server utilization, thus lower costs.
- The threats caused by multi-tenancy differ from one cloud delivery model to another.

Legal protection of cloud users

- The contract between the user and the Cloud Service Provider (CSP) should spell out explicitly:
  - CSP obligations to handle securely sensitive information and its obligation to comply to privacy laws.
  - CSP liabilities for mishandling sensitive information.
  - CSP liabilities for data loss.
  - The rules governing ownership of the data.
  - The geographical regions where information and backups can be stored.

Operating system security

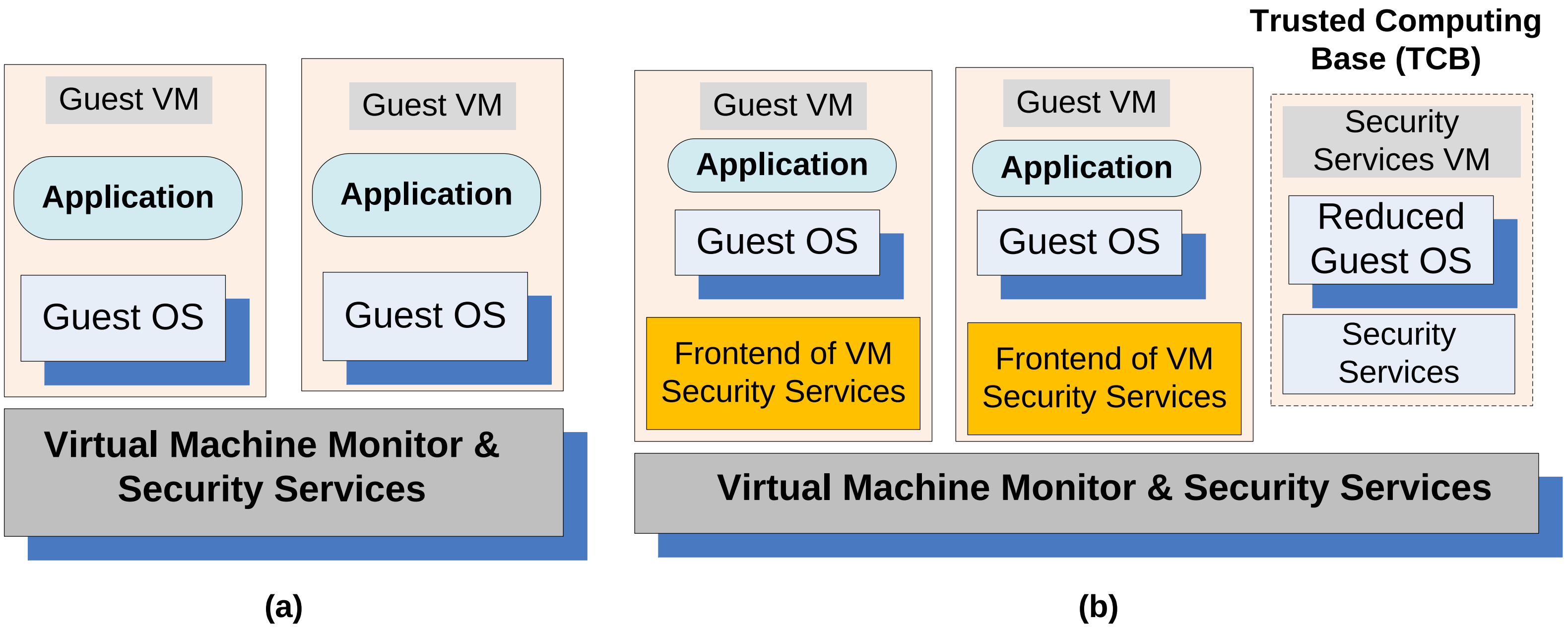
- A critical function of an OS is to protect applications against a wide range of malicious attacks e.g., unauthorized access to privileged information, tempering with executable code, and spoofing.
- The elements of the mandatory OS security:
  - Access control → mechanisms to control the access to system objects.
  - Authentication usage → mechanisms to authenticate a principal.
  - Cryptographic usage policies → mechanisms used to protect the data
- Commercial OS do not support a multi-layered security; only distinguish between a completely privileged security domain and a completely unprivileged one.
- Trusted paths mechanisms → support user interactions with trusted software. Critical for system security; if such mechanisms do not exist, then malicious software can impersonate trusted software. Some systems provide trust paths for a few functions such as login authentication and password changing and allow servers to authenticate their clients.

Closed-box versus open-box platforms

- Closed-box platforms e.g., cellular phones, game consoles and ATM could have embedded cryptographic keys to reveal their true identity to remote systems and authenticate the software running on them.
- Such facilities are not available to open-box platforms, the traditional hardware for commodity operating systems.
- Commodity operating system offer low assurance. An OS is a complex software system consisting of millions of lines of code and it is vulnerable to a wide range of malicious attacks.
- An OS provides weak mechanisms for applications to authenticate to one another and create a trusted path between users and applications.
- An OS poorly isolates one application from another, once an application is compromised, the entire physical platform and all applications running on it can be affected. The platform security level is reduced to the security level of the most vulnerable application running on the platform.

Virtual machine security

- Hybrid and hosted VMs, expose the entire system to the vulnerability of the host OS.
- In a traditional VM the Virtual Machine Monitor (VMM) controls the access to the hardware and provides a stricter isolation of VMs from one another than the isolation of processes in a traditional OS.
  - A VMM controls the execution of privileged operations and can enforce memory isolation as well as disk and network access.
  - The VMMs are considerably less complex and better structured than traditional operating systems thus, in a better position to respond to security attacks.
  - A major challenge → a VMM sees only raw data regarding the state of a guest operating system while security services typically operate at a higher logical level, e.g., at the level of a file rather than a disk block.
- A secure TCB (Trusted Computing Base) is a necessary condition for security in a virtual machine environment; if the TCB is compromised then the security of the entire system is affected.


Trusted Computing Base (TCB)
Guest VM
Application
Guest OS
Guest VM
Application
Guest OS
Virtual Machine Monitor & Security Services
(a)
Guest VM
Application
Guest OS
Frontend of VM Security Services
Guest VM
Application
Guest OS
Frontend of VM Security Services
Security Services VM
Reduced Guest OS
Security Services
Virtual Machine Monitor & Security Services
(b)

(a) Virtual security services provided by the VMM; (b) A dedicated security VM.

VMM-based threats

- Starvation of resources and denial of service for some VMs. Probable causes:
  - (a) badly configured resource limits for some VMs;
  - (b) a rogue VM with the capability to bypass resource limits set in VMM.
- VM side-channel attacks: malicious attack on one or more VMs by a rogue VM under the same VMM. Probable causes:
  - (a) lack of proper isolation of inter-VM traffic due to misconfiguration of the virtual network residing in the VMM;
  - (b) limitation of packet inspection devices to handle high speed traffic, e.g., video traffic;
  - (c) presence of VM instances built from insecure VM images, e.g., a VM image having a guest OS without the latest patches.
- Buffer overflow attacks.

VM-based threats

- Deployment of rogue or insecure VM; unauthorized users may create insecure instances from images or may perform unauthorized administrative actions on existing VMs. Probable cause:
  - improper configuration of access controls on VM administrative tasks such as instance creation, launching, suspension, re-activation and so on.
- Presence of insecure and tampered VM images in the VM image repository. Probable causes:
  - (a) lack of access control to the VM image repository;
  - (b) lack of mechanisms to verify the integrity of the images, e.g., digitally signed image.

Security of virtualization

- The complete state of an operating system running under a virtual machine is captured by the VM; this state can be saved in a file and then the file can be copied and shared. Implications:
  - Ability to support the IaaS delivery model; in this model a user selects an image matching the local environment used by the application and then uploads and runs the application on the cloud using this image.
  - Increased reliability; an operating system with all the applications running under it can be replicated and switched to a hot standby
  - Improved intrusion prevention and detection; a clone can look for known patterns in system activity and detect intrusion. The operator can switch to a hot standby when suspicious events are detected.
  - More efficient and flexible software testing; instead of very large number of dedicated systems running under different OS, different version of each OS, and different patches for each version, virtualization allows the multitude of OS instances to share a small number of physical systems.

More advantages of virtualization

- Straightforward mechanisms to implement resource management policies:
  - To balance the load of a system, a VMM can move an OS and the applications running under it to another server when the load on the current server exceeds a high water mark.
  - To reduce power consumption the load of lightly loaded servers can be moved to other servers and then turn off or set on standby mode the lightly loaded servers.
- When secure logging and intrusion protection are implemented at the VMM layer, the services cannot be disabled or modified; intrusion detection can be disabled and logging can be modified by an intruder when implemented at the OS level. A VMM may be able to log only events of interest for a post-attack analysis.

Undesirable effects of virtualization

- Diminished ability to manage the systems and track their status.
  - The number of physical systems in the inventory of an organization is limited by cost, space, energy consumption, and human support. Creating a virtual machine (VM) reduces ultimately to copying a file, therefore the explosion of the number of VMs. The only limitation for the number of VMs is the amount of storage space available.
  - Qualitative aspect of the explosion of the number of VMs → traditionally, organizations install and maintain the same version of system software. In a virtual environment the number of different operating systems, their versions, and the patch status of each version will be very diverse. Heterogeneity will tax the support team.
  - The software lifecycle has serious implication on security. The traditional assumption → the software lifecycle is a straight line, hence the patch management is based on a monotonic forward progress. The virtual execution model maps to a tree structure rather than a line; indeed, at any point in time multiple instances of the VM can be created and then each one of them can be updated, different patches installed, and so on.

Implications of virtualization on security

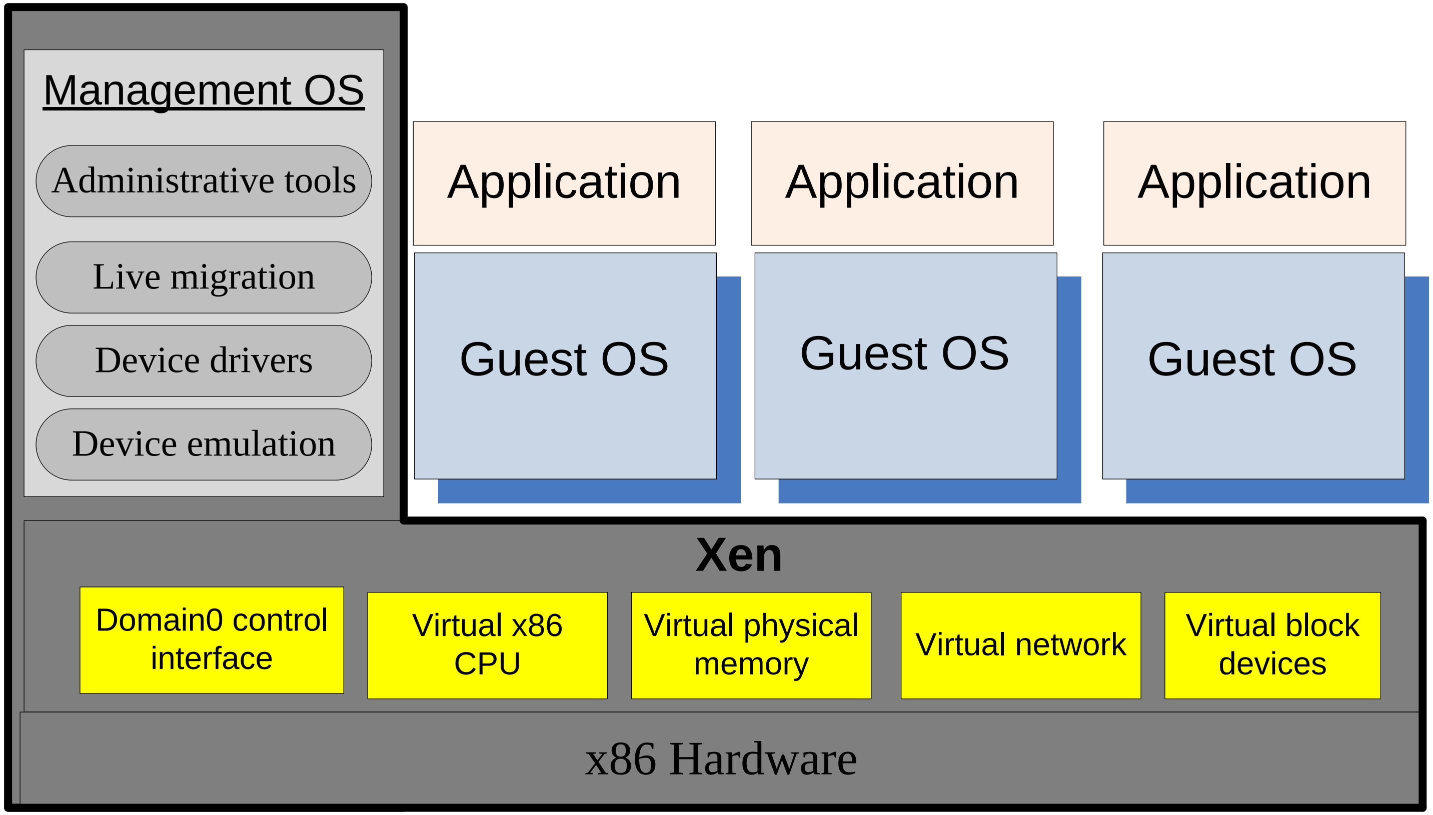
- Infection may last indefinitely → some of the infected VMs may be dormant at the time when the measures to clean up the systems are taken and then, at a later time, wake up and infect other systems; the scenario can repeat itself.
- In a traditional computing environment a steady state can be reached. In this steady state all systems are brought up to a desirable state. This desirable state is reached by installing the latest version of the system software and then applying to all systems the latest patches. Due to the lack of control, a virtual environment may never reach such a steady state.
- A side effect of the ability to record in a file the complete state of a VM is the possibility to roll back a VM. This allows a new type of vulnerability caused by events recorded in the memory of an attacker.
- Virtualization undermines the basic principle that time sensitive data stored on any system should be reduced to a minimum.

Security risks posed by shared images

- Image sharing is critical for the IaaS cloud delivery model. For example, a user of AWS has the option to choose between
  - Amazon Machine Images (AMIs) accessible through the Quick Start.
  - Community AMI menus of the EC2 service.
- Many of the images analyzed by a recent report allowed a user to undelete files, recover credentials, private keys, or other types of sensitive information with little effort and using standard tools.
- A software vulnerability audit revealed that 98% of the Windows AMIs and 58% of Linux AMIs audited had critical vulnerabilities.
- Security risks:
  - Backdoors and leftover credentials.
  - Unsolicited connections.
  - Malware.

Security risks posed by a management OS

- A virtual machine monitor or hypervisor is considerably smaller than an operating system e.g., the Xen VMM has ~ 60,000 lines of code.
- The Trusted Computer Base (TCB) of a cloud computing environment includes not only the hypervisor but also the management OS.
- The management OS supports administrative tools, live migration, device drivers, and device emulators.
- In Xen the management operating system runs in Dom0; it manages the building of all user domains, a process consisting of several steps:
  - Allocate memory in the Dom0 address space and load the kernel of the guest operating system from the secondary storage.
  - Allocate memory for the new VM and use foreign mapping to load the kernel to the new VM.
  - Set up the initial page tables for the new VM.
  - Release the foreign mapping on the new VM memory, set up the virtual CPU registers and launch the new VM.



The trusted computing base of a Xen-based environment includes the hardware, Xen, and the management operating system running in Dom0. The management OS supports administrative tools, live migration, device drivers, and device emulators. A guest operating system and applications running under it reside in a DomU.

Possible actions of a malicious Dom0

- At the time it creates a DomU:
  - Refuse to carry out the steps necessary to start the new VM.
  - Modify the kernel of the guest OS to allow a third party to monitor and control the execution of applications running under the new VM.
  - Undermine the integrity of the new VM by setting the wrong page tables and/or setup wrong virtual CPU registers.
  - Refuse to release the foreign mapping and access the memory while the new VM is running.
- At run time:
  - Dom0 exposes a set of abstract devices to the guest operating systems using split drivers with the frontend of in a DomU and the backend in Dom0. We have to ensure that run time communication through Dom0 is encrypted. Transport Layer Security (TLS) does not guarantee that Dom0 cannot extract cryptographic keys from the memory of the OS and applications running in DomU

A major weakness of Xen

- The entire state of the system is maintained by XenStore.
- A malicious VM can deny to other VMs access to XenStore; it can also gain access to the memory of a DomU.

How to deal with run-time vulnerability of Dom0

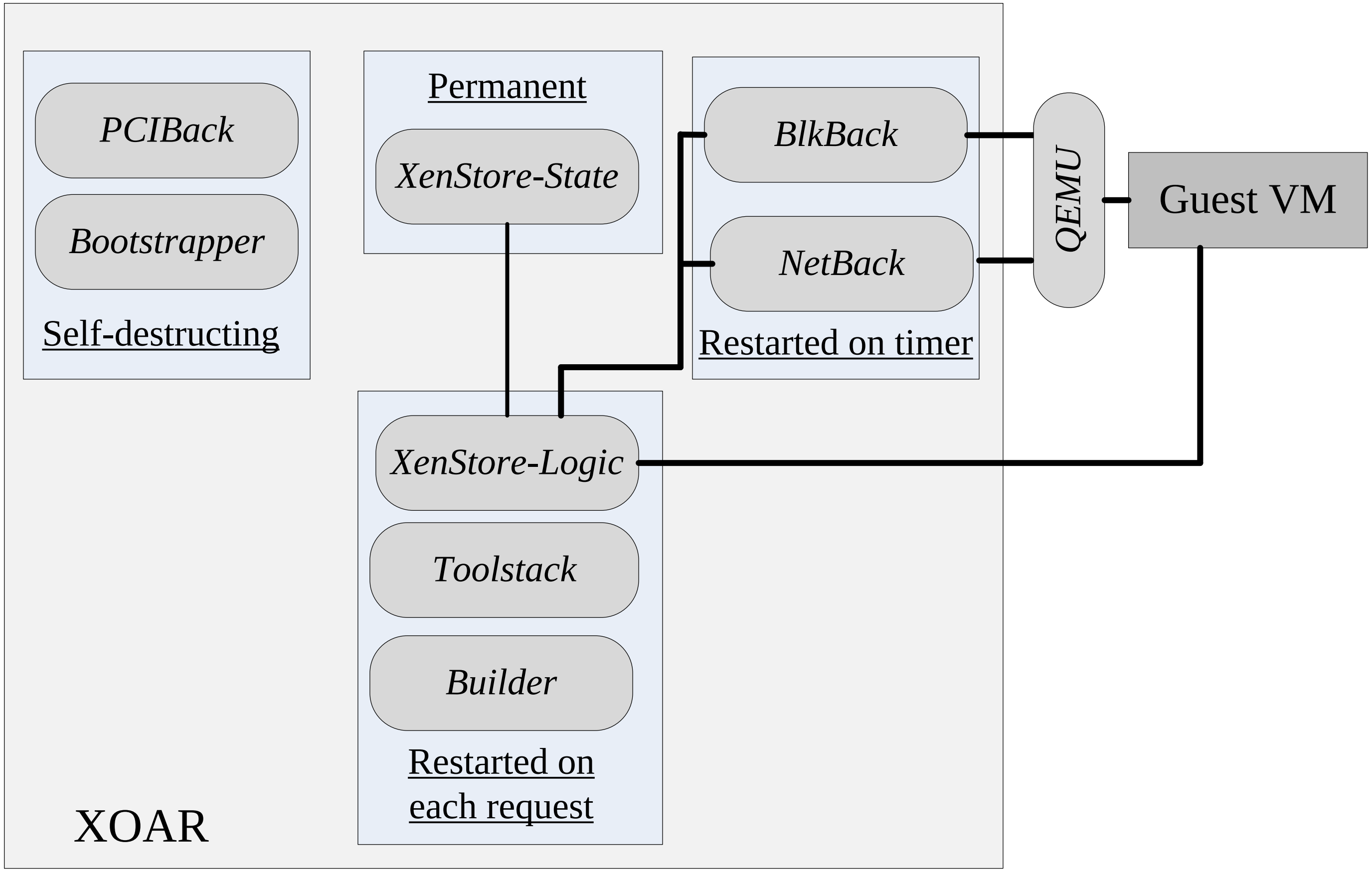
- To implement a secure run-time system we have to intercept and control the hypercalls used for communication between a Dom0 that cannot be trusted and a DomU we want to protect.
- New hypercalls are necessary to protect:
  - The privacy and integrity of the virtual CPU of a VM. When Dom0 wants to save the state of the VM the hypercall should be intercepted and the contents of the virtual CPU registers should be encrypted. When DomU is restored the virtual CPU context should be decrypted and then an integrity check should be carried out.
  - The privacy and integrity of the VM virtual memory. The page table update hypercall should be intercepted and the page should be encrypted so that Dom0 handles only encrypted pages of the VM. To guarantee the integrity the hypervisor should calculate a hash of all the memory pages before they are saved by Dom0. An address translation is necessary as a restored DomU may be allocated a different memory region.
  - The freshness of the virtual CPU and the memory of the VM. The solution is to add to the hash a version number.

Xoar - breaking the monolithic design of TCB

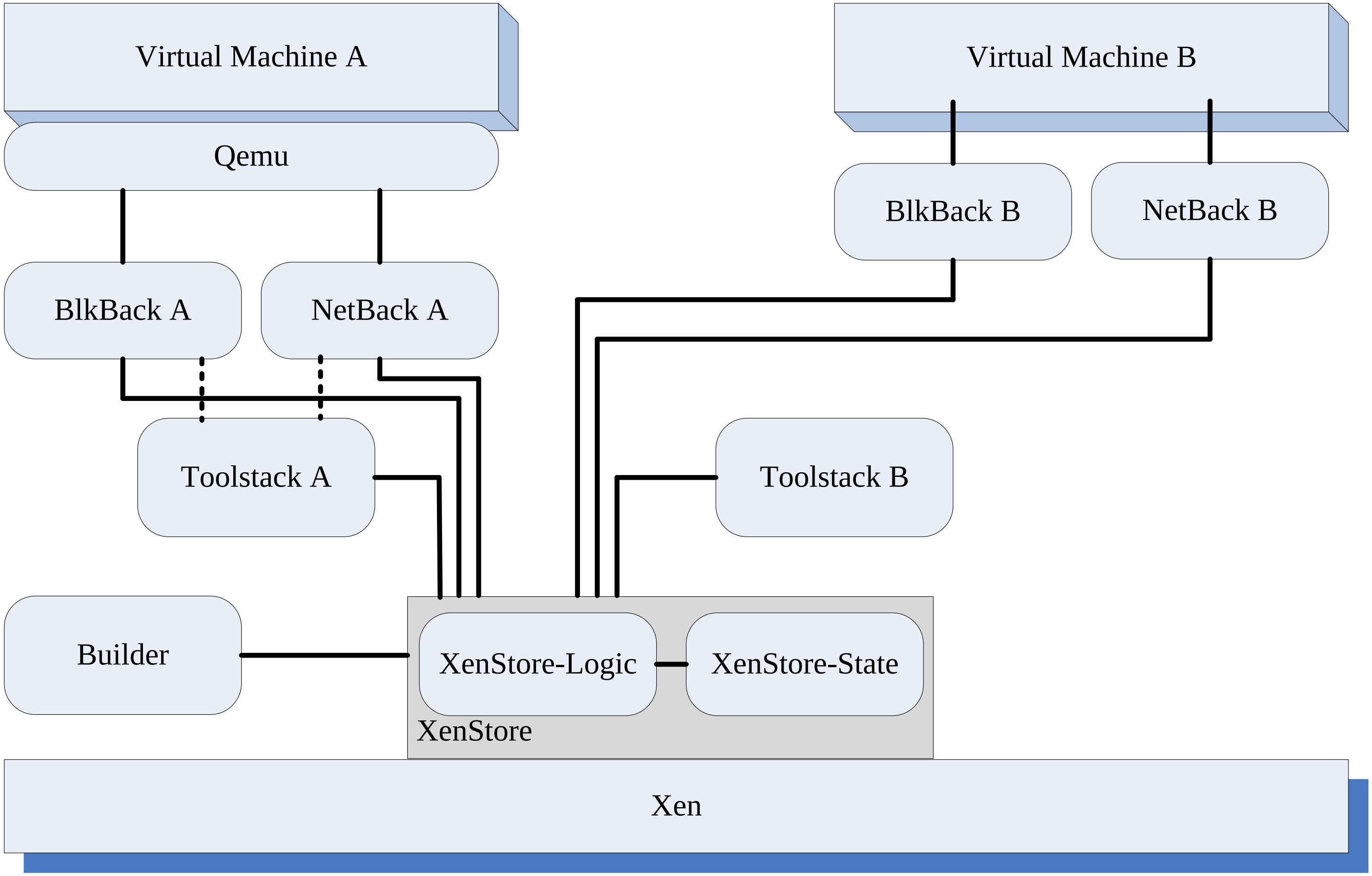
- Xoar is a version on Xen designed to boost system security; based on micro-kernel design principles. The design goals are:
  - Maintain the functionality provided by Xen.
  - Ensure transparency with existing management and VM interfaces.
  - Tight control of privileges, each component should only have the privileges required by its function.
  - Minimize the interfaces of all components to reduce the possibility that a component can be used by an attacker.
  - Eliminate sharing. Make sharing explicit whenever it cannot be eliminated to allow meaningful logging and auditing.
  - Reduce the opportunity of an attack targeting a system component by limiting the time window when the component runs.
- The security model of Xoar assumes that threats come from:
  - A guest VM attempting to violate data integrity or confidentiality of another guest VM on the same platform, or to exploit the code of the guest.
  - Bugs in initialization code of the management virtual machine.

Xoar system components

- Permanent components → XenStore-State} maintains all information regarding the state of the system.
- Components used to boot the system; they self-destruct before any user VM is started. They discover the hardware configuration of the server including the PCI drivers and then boot the system:
  - PCIBack - virtualizes access to PCI bus configuration.
  - Bootstrapper - coordinates booting of the system.
- Components restarted on each request:
  - XenStore-Logic
  - Toolstack - handles VM management requests, e.g., it requests the Builder to create a new guest VM in response to a user request.
  - Builder - initiates user VMs.
- Components restarted on a timer: the two components export physical storage device drivers and the physical network driver to a guest VM.
  - Blk-Back - exports physical storage device drivers using udev rules.
  - NetBack - exports the physical network driver.



Xoar has nine classes of components of four types: permanent, self-destructing, restarted upon request, and restarted on timer. A guest VM is started using the by the Builder using the Toolstack; it is controlled by the XenStore-Logic. The devices used by the guest VM are emulated by the Qemu component. Qemu is responsible for device emulation



Component sharing between guest VM in Xoar. Two VM share only the XenStore components. Each one has a private version of the BlkBack, NetBack and Toolstack.