Felhő alapú rendszerek

Kovács László, ME

Kliens-szerver architektúra

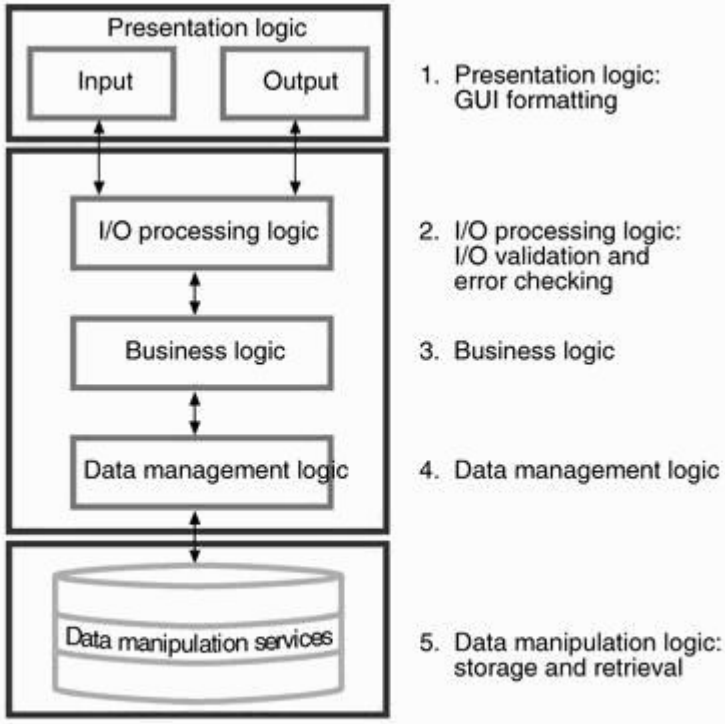
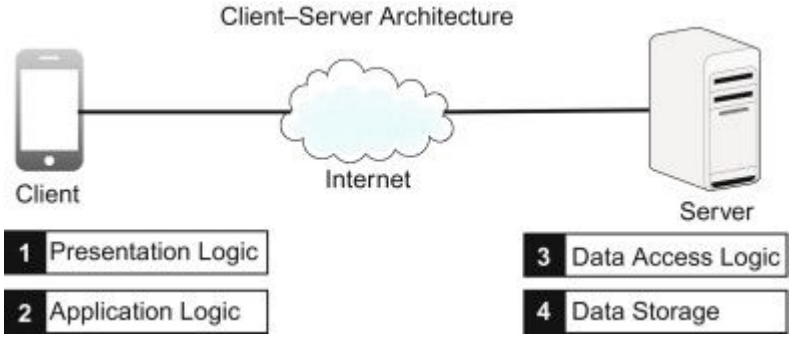


FIGURE 12.15 APPLICATION FUNCTIONAL LOGIC COMPONENTS



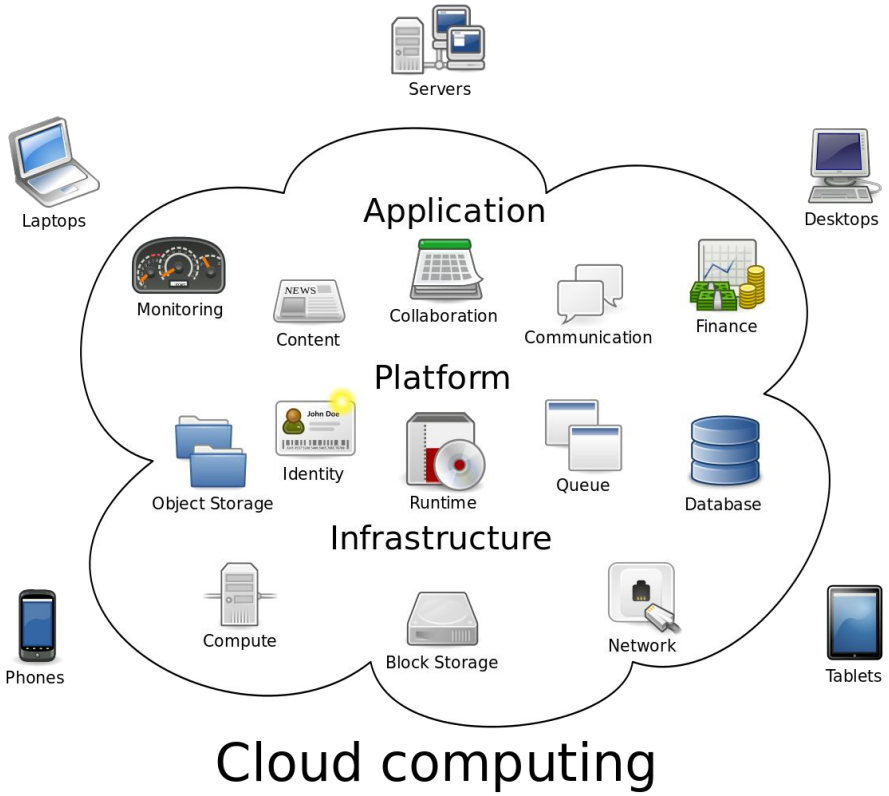
Lokális szerverek:

- Összetett rendszerek
- Nagyobb karbantartási költség
- Nagyobb helyi felelősség

Felhő alapú architektúra

Cloud computing:

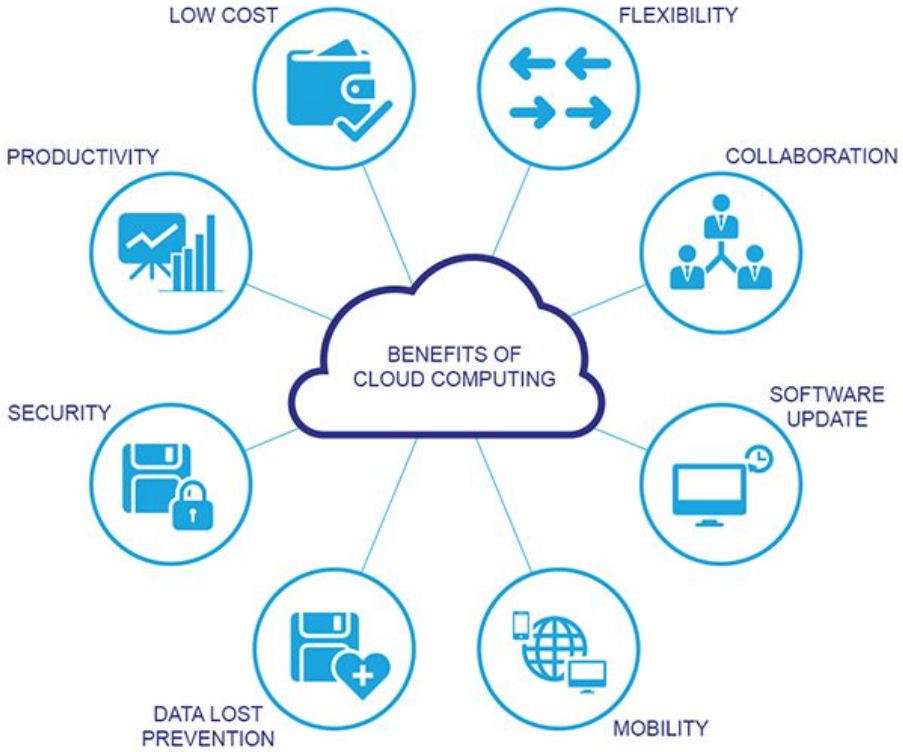
- Technológia modell
- Üzleti, működési modell
- Az alkalmazás a szervereken futó szolgáltatás komponensekből áll össze
- Sok szerver együttese
- Nem, fix telepítésű szerverek
- Külső karbantartás



Felhő alapú architektúra

Cloud computing előnyei:

- Skálázható
- Igény szerinti erőforrás használat
- Rugalmasság
- Védelem
- Erőforrás, felügyelet kiszervezés
- Internet alapú
- Centralizált felügyelet
- Hatékonyság



Grid Computing

Solving large problems with parallel computing

Made mainstream by Globus Alliance

Utility Computing

Offering computing resources as a metered service

Introduced in late 1990s

Software as a Service

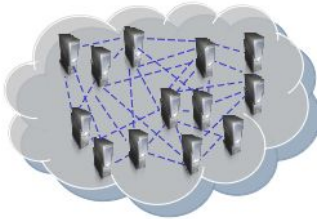
Network-based subscriptions to applications

Gained momentum in 2001

Cloud Computing

Next-Generation Internet computing

Next-Generation Data Centers

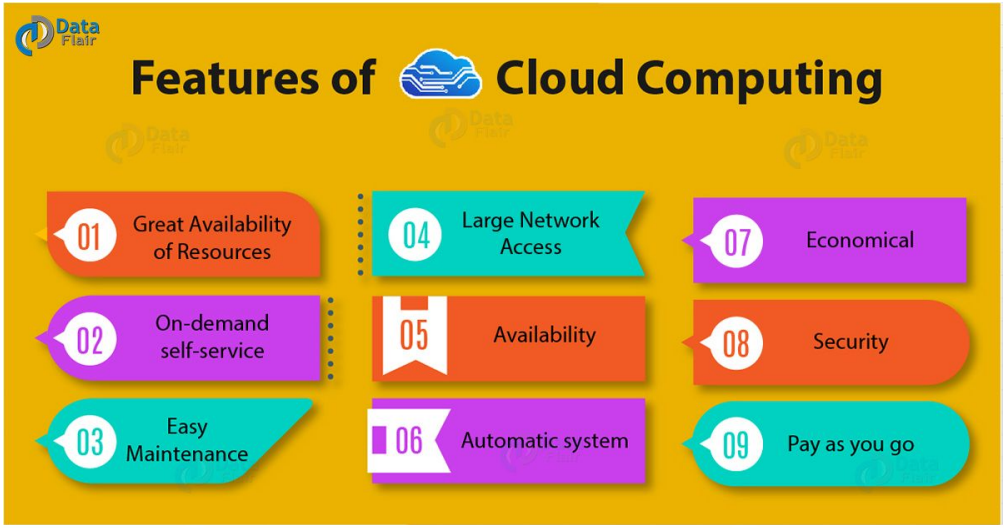



Scalability
Elasticity
• Dynamic provision
• Multi-tenant design
Availability
Reliability
• Fault tolerance
• System resilience
• System security
Performance
Optimization
• Parallel processing
• Load balancing
• Job scheduling
Accessibility
Portability
• Uniform access
• Thin client
Manageability
Interoperability
• Control automation
• System monitoring
• Billing system
Utility
Computing
SOA + SLA

Felhő alapú architektúra

Cloud computing alapigényei:

- Gyors hálózat
- Szakértelem
- Elosztott szerverek
- Virtualizációs rendszerek
- Open source rendszerek



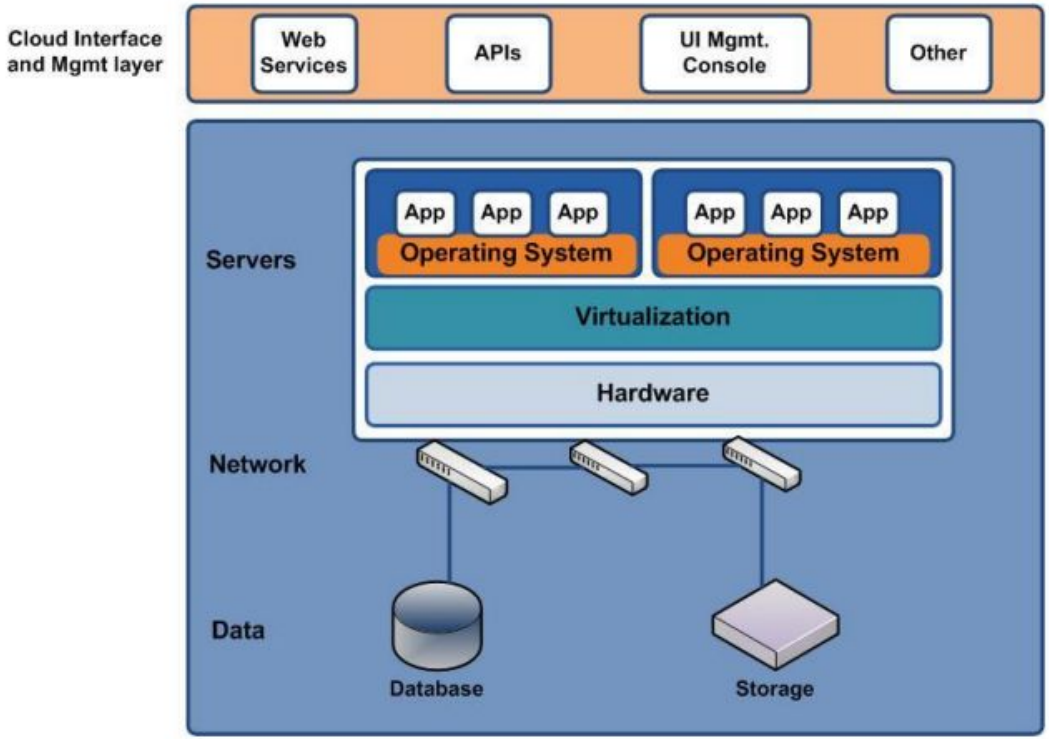
	Medium-sized DC	Very Large DC	Ratio (Large-to-Small DC)
Network	$95 per Mbit/sec/month	$13 per Mbit/sec/month	7.1
Storage	$2.20 per GByte/month	$0.40 per GByte/month	5.7
Administration	~140 Servers/ Admin	>1000 Servers/ Admin	7.1

Table 1: Economies of scale in 2006 for a medium-sized datacenter (1000 servers) vs. a

Virtualizáció

Virtuális környezet:

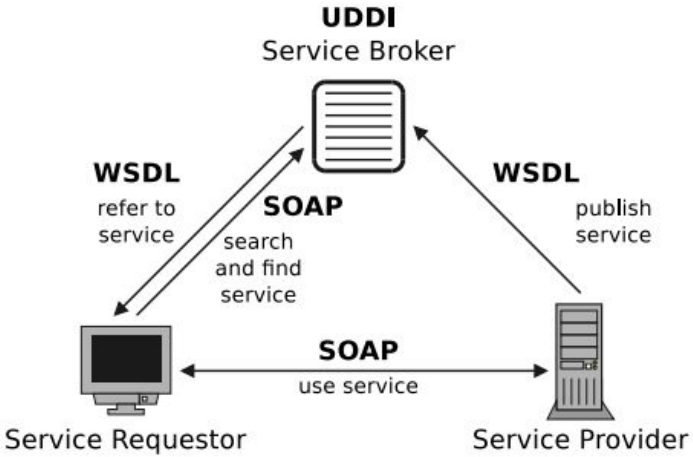
- OS: operációs rendszer
- vendég OS a gazda OS felett
- több vendég OS is lehet
- Zártság
- Védettség
- Közös eszközhasználat



Docker Container és VM

SOA: szolgáltatás alapú architektúra

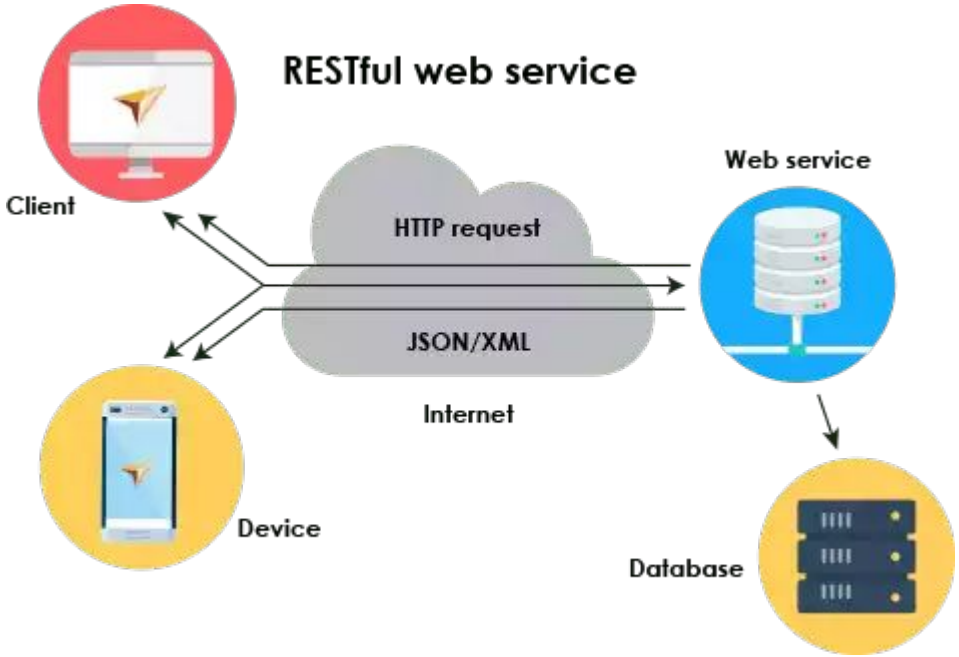
- SOA is a method to encapsulate IT components in services and coordinate (*orchestrate*) them
- Objective: Grouping of services to higher-level services and simple provisioning to other organizational departments or customers
- Theory: Software development costs can be reduced in the long term
  - Reason: In the future, eventually, all required services are available and they only need to be orchestrated
- Technical implementation: **Web services**



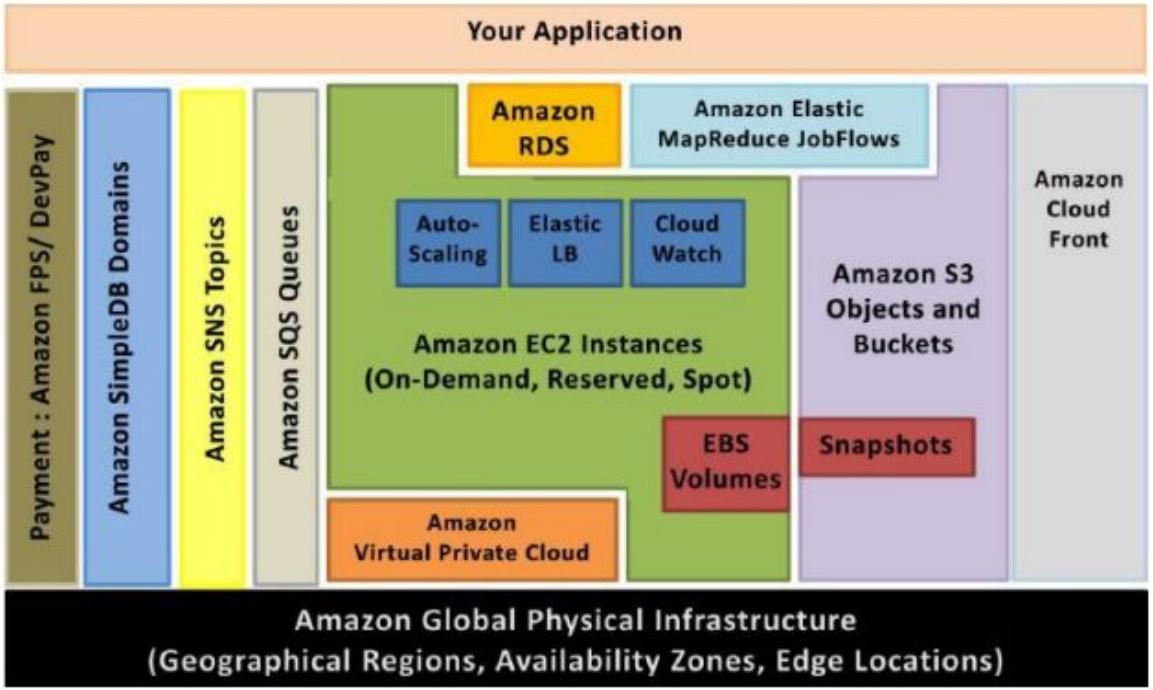
Felhő szolgáltatások

WEB szolgáltatás alapú:

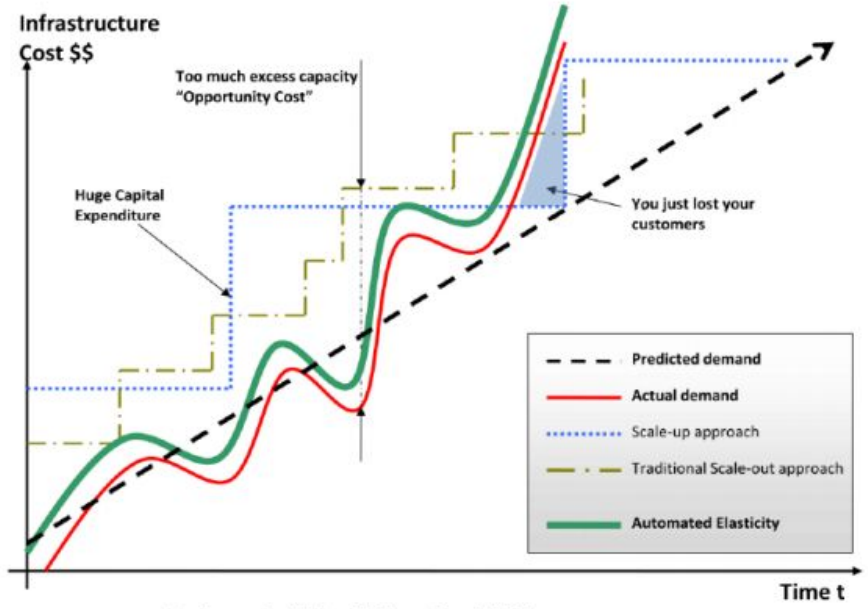
- kliens megszólítja a modul gazdát
- elküldi a feladatot egy üzenetben
- a feladat várakozósorba kerül
- végrehajtás után üzenet vissza a klienshez az eredmény adatokkal
- egy modulnak több gazdája is lehet



Felhő szolgáltatások



1. **Vertical scaling**: purchase two powerful servers, $50,000/each. This should provide the capacity to host up to 50,000 users. When these servers reach capacity, XYZ will buy another two servers.
2. **Horizontal scaling**: purchase two servers, $3,000/ each, with the capacity to serve 10,000 users. When XYZ needs to serve more users, they will buy additional servers.
3. **Automated Scaling:** purchase the required computing resources from a cloud provider to serve 5,000 users, and let the cloud provider auto-scale the capacity.



Automated Elasticity + Scalability

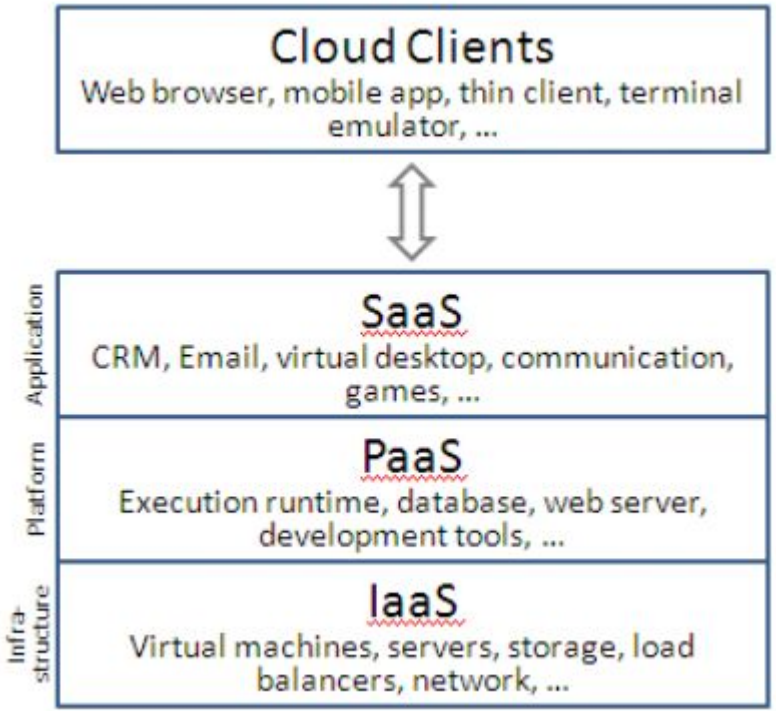
Felhő szolgáltatások

Szolgáltatási módok

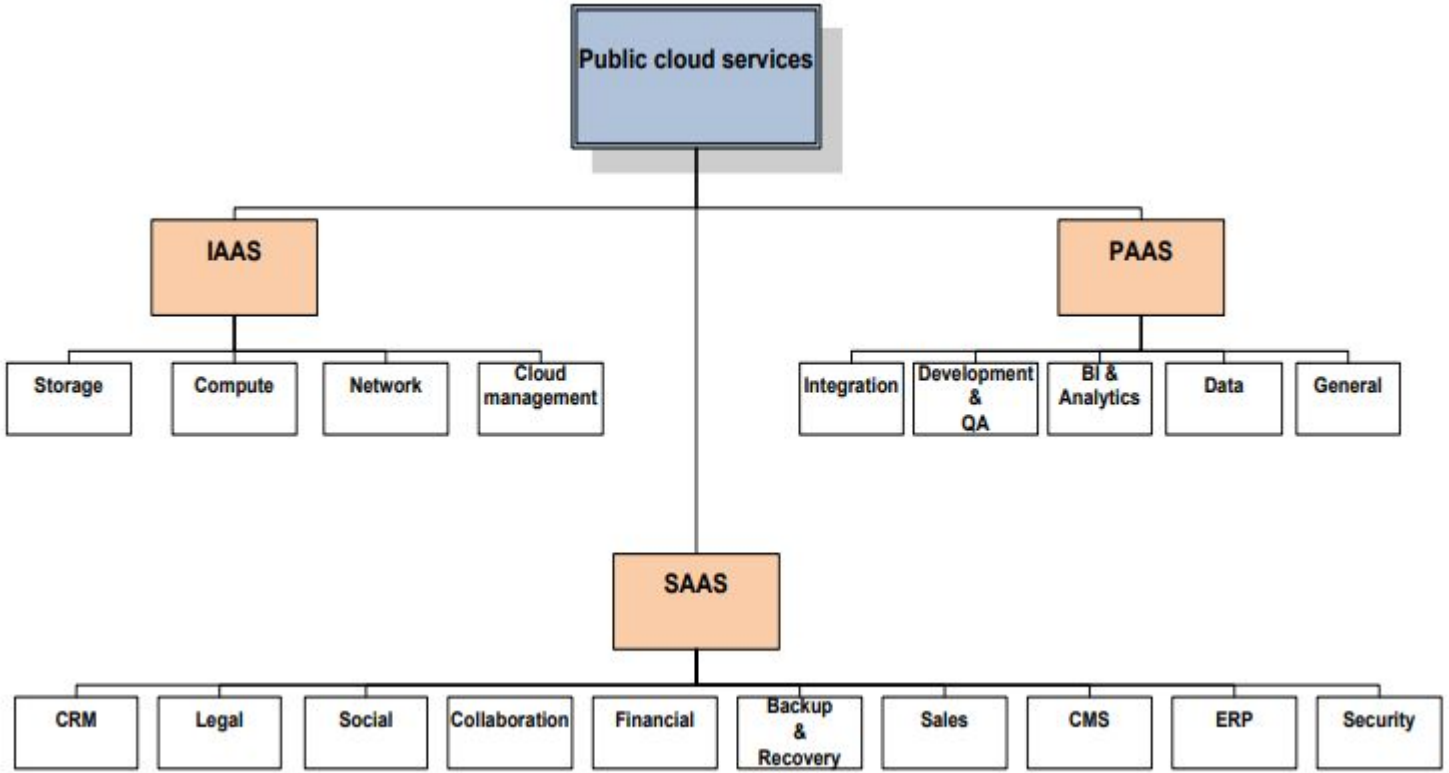
IAAS: OS és eszközhasználat , viszonylag alacsony szinten

PAAS: szerver szintű alkalmazások használatának biztosítása

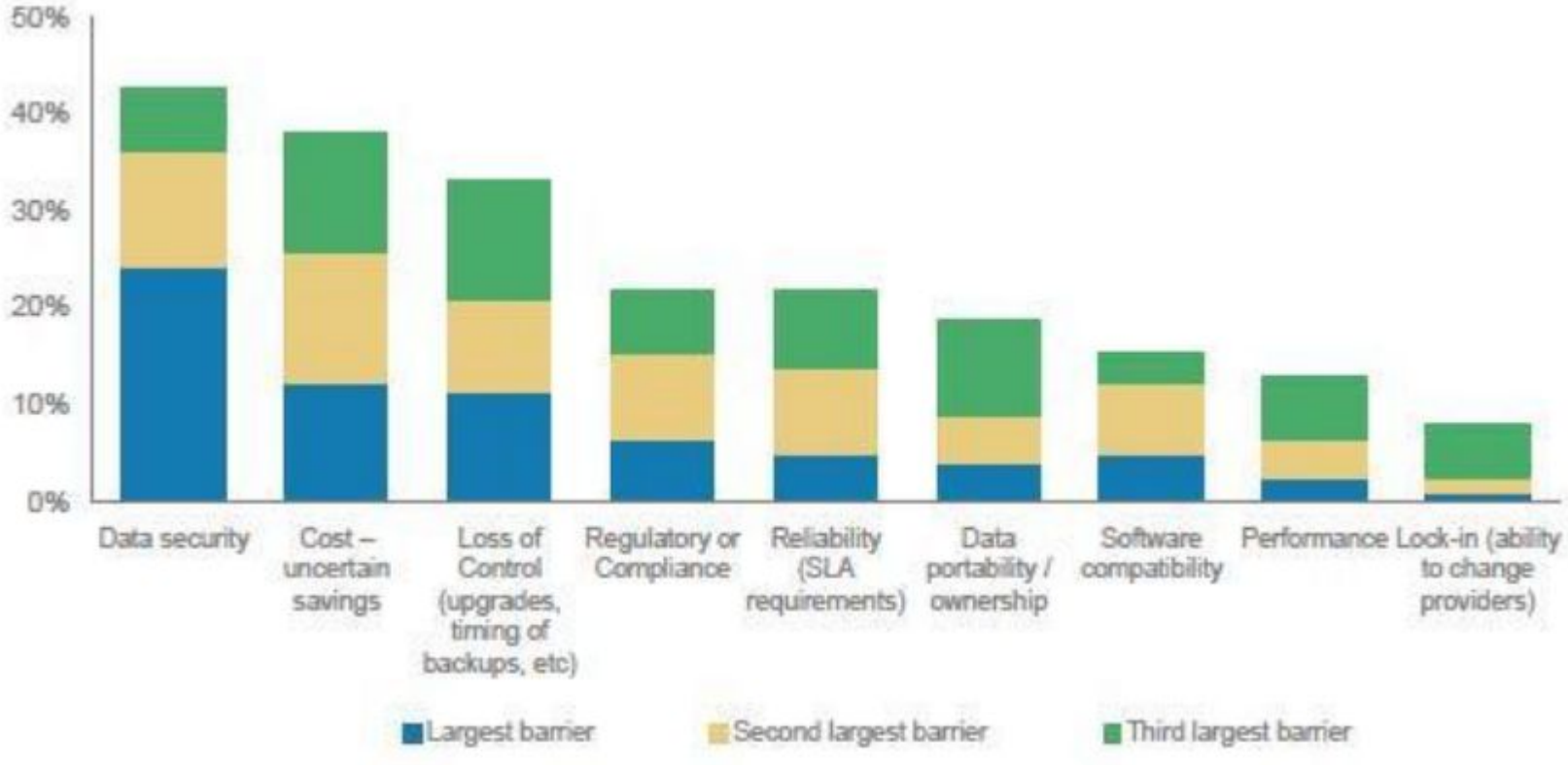
SAAS: kész üzleti alkalmazások használatának biztosítása



Felhő szolgáltatások



Felhő architektúra korlátai



Felhő alapú adattárolás

Adattárolás jellemzői

- megbízhatóság
- redundancia
- központi felügyelet
- skálázhatóság
- gazdag API készlet
-

	Amazon S3	Rackspace CloudFiles
Storage	0.11/GB/month	0.10/GB/month
Data Transfer IN	Free	Free
Data Transfer OUT	0.09/GB/month	0.18/GB/month
HTTP requests (PUT, COPY, POST)	0.01/1000 requests	Free
HTTP requests (GET, HEAD)	0.01/10000 requests	Free

Felhő alapú adattárolás

- Who has access to the data? What are the access-control policies? Do I have full visibility into information regarding these access-control policies?
- Is data encrypted during transfer from the internal network to the public cloud? What is the encryption algorithm? Can data be encrypted when stored in the cloud? Who holds the encryption keys?

  If a cloud provider is not supposed to have access to the data, encryption keys should be held only by the company that owns the data. Some of the compliance standards mandate full data encryption, and do not permit cloud providers to hold encryption keys.
- What is the disaster-recovery process? Does the cloud provider replicate data across multiple datacenters? Are these datacenters located in different geographical locations?

  If data is stored in only one datacenter and the cloud provider doesn't have the capability to replicate it at other datacenters, this should raise a red flag.
- What is the data-backup process? Who has access to the backup data? Where is the backup data stored?
- What is the data-recovery process? How long does data recovery take?
- What is the security-breach investigation process? Does the cloud provider have security-breach investigation capabilities?

  This question is often forgotten, but it is very important – if data is compromised, the cloud provider will be the only source of information for any investigation.

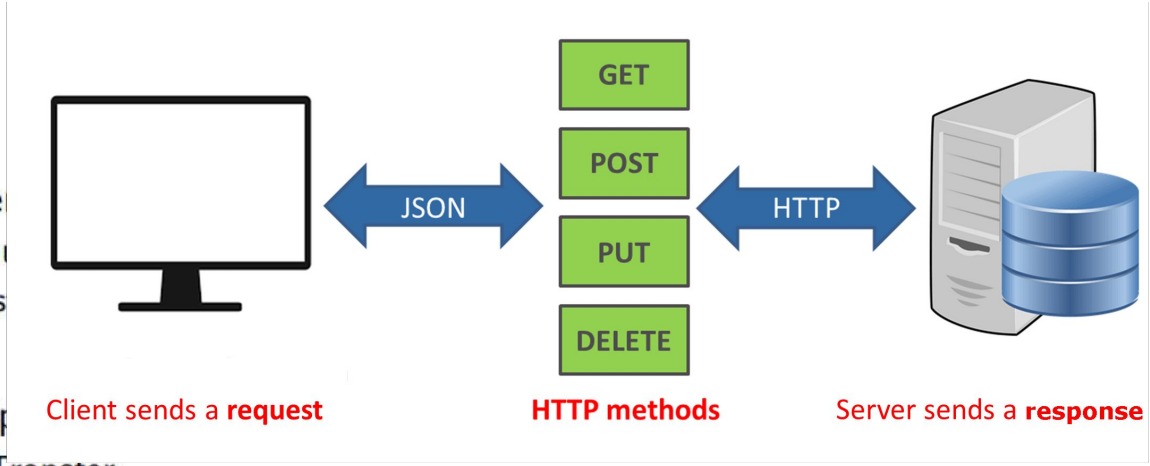
Felhő alapú szoftver szolgáltatás

- Fact: most virtualized environments have highly variable performance
- Variance also due to multi-tenancy, movements of large amounts of data, and the system itself (e.g., HDFS randomly distributes data blocks across a cluster)
- Even if CPU and memory sharing is not a problem, I/O sharing could easily kill performance
- Many HPC applications need to ensure that all the threads of a program are running simultaneously
- **RQ: How to make performance more predictable?**
- **RQ: How to guarantee performance/QoS?**

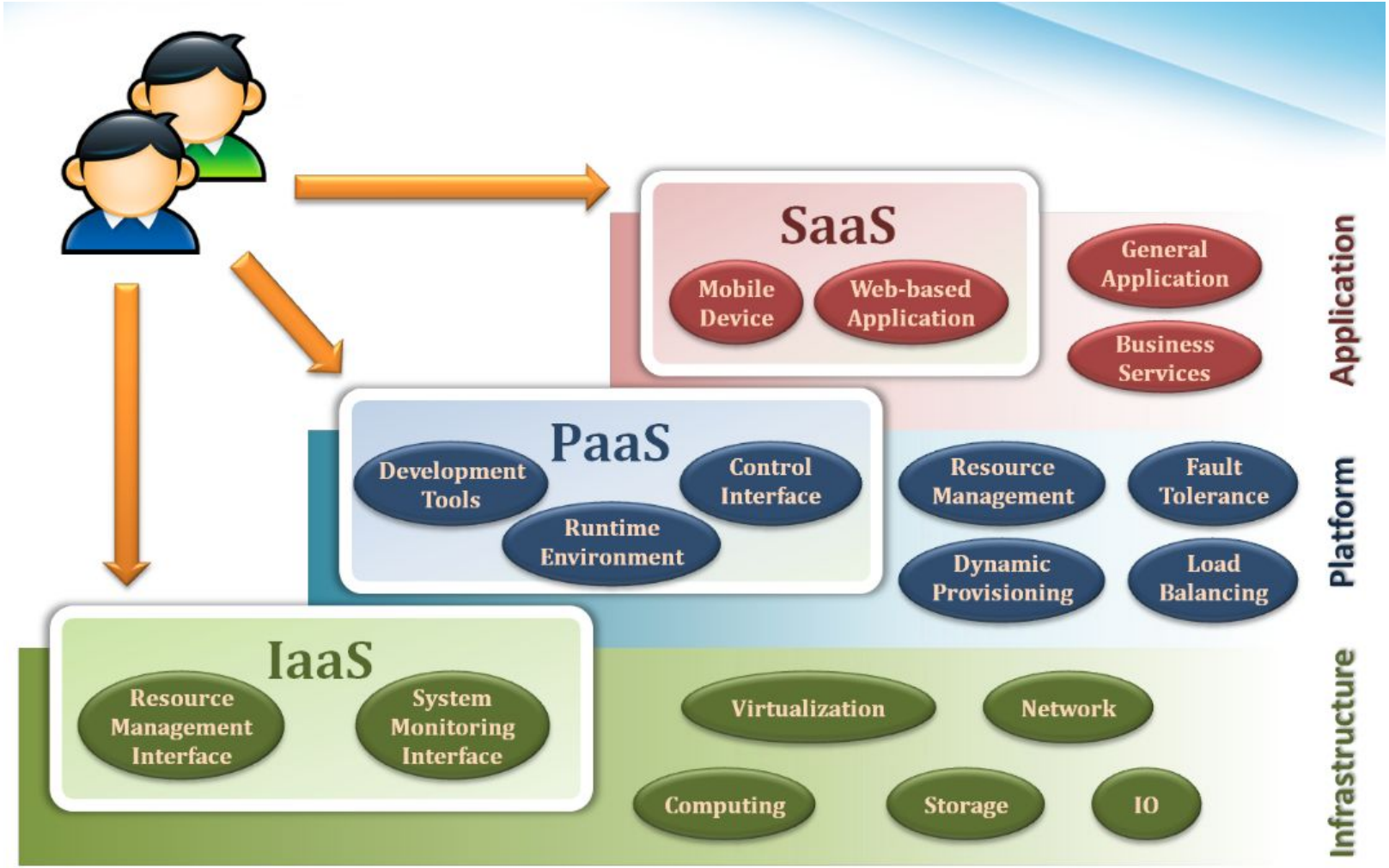
- Scalability is key: quick, automatic scale up or down according to user's changing needs
- Application's scalability is another issue
  - 1 machine x 100 hrs = 100 machines x 1 hr?
- Ideally, you pay as you go, and are charged by the cycles (compute), or the bytes (storage and communication)
- **RQ: How to predict and react to workload changes quickly and dynamically?**
- **RQ: How to reduce bottlenecks and provide for the best speedups?**
- **RQ: How to charge more accurately and fairly?**
- **RQ: How to scale data storages?**

REST API

- Communication via RESTful web se
  - REST implements stateless commu
    - The server does not store any s
- SOAP uses XML for requests
- With REST, a transfer of a state rep
  - REST = REpresentational State Transfer
- The 4 HTTP methods `PUT`, `GET`, `POST` and `DELETE` are sufficient to initiate all necessary functions on objec
  - Create record
  - Read record
  - Update record
  - Erase record



HTTP	CRUD Actions	SQL	Description
`PUT/POST`	Create	`INSERT`	Create or replace a resource
`GET`	Read/Retrieve	`SELECT`	Request a resource
`PUT`	Update	`UPDATE`	Modify a resource
`DELETE`	Delete/Destroy	`DELETE`	Erase a resource


SaaS
Mobile Device
Web-based Application
General Application
Business Services
PaaS
Development Tools
Control Interface
Runtime Environment
Resource Management
Fault Tolerance
Dynamic Provisioning
Load Balancing
IaaS
Resource Management Interface
System Monitoring Interface
Virtualization
Network
Computing
Storage
IO
Application
Platform
Infrastructure