Cloud computing

- La definizione classica di riferimento è quella del National Institute of Standards and Technology (NIST) USA (http://goo.gl/eBGBk)
- In sintesi il Cloud computing si occupa di:

Fornitura di tecnologia di informazione e comunicazione (ICT) come servizio

Caratteristiche del Cloud

- **Self-service, on-demand**
  - Il cliente chiede autonomamente ciò che gli serve, quando gli serve (e sperabilmente lo ottiene).
- **Accesso attraverso la rete**
  - Assume che una rete (Internet o intranet) sia disponibile, normalmente a banda larga.
- **Pool di risorse**
  - L'utente non si preoccupa di conoscere i dettagli delle risorse, che sono gestiti dai Cloud resource provider.
- **Elasticità**
  - Il servizio Cloud può scalare rapidamente come dimensioni a seconda delle necessità del cliente.
- **Pagamento a consumo**
  - Il cliente paga solo per ciò che usa.

Una analogia: l'autonoleggio

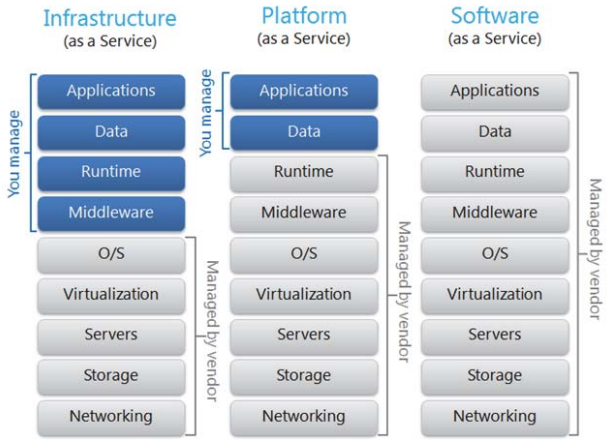
- Self-service, on-demand
  - Prenotazione telefonica oppure online
- Rete
  - Estesa rete di autonoleggi in tutto il mondo
- Pool di risorse
  - Pensa l'autonoleggio a gestire sapere quante macchine gli servono
- Elasticità
  - Il numero di auto disponibili normalmente varia a seconda della richiesta
- Pagamento a consumo
  - Il cliente paga per il tempo in cui usa l'auto (e non pensa ad assicurazione, gomme, etc.)

Fonte: http://goo.gl/cEa8M

Il focus sul "service"

- Abbiamo visto che nella definizione di Cloud computing ("Fornitura di tecnologia di informazione e comunicazione come servizio") il ***servizio*** nei confronti del cliente è parte essenziale.
- Il Cloud computing si può modellare infatti intorno a *servizi* legati principalmente a
  - Infrastruttura (**IaaS** → Infrastructure as a Service)
  - Piattaforma (**PaaS** → Platform as a Service)
  - Software (**SaaS** → Software as a Service)

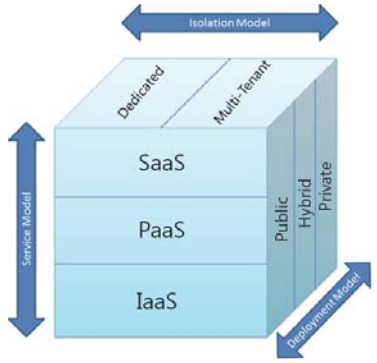
Chi fa cosa?



Fonte: http://goo.gl/1jmkR

Aggiungiamo dimensioni

- Oltre i modelli di *servizio*, parti importanti per definire e capire il Cloud computing sono i modelli di:
  - ***deployment*** (dove distribuisco i servizi)
  - ***isolamento*** (come isolo i servizi)



Fonte: http://goo.gl/1jmkR

Deployment: i "tipi di Cloud"

- **Cloud privata:**
  - L'infrastruttura viene fornita per un *uso* esclusivo da parte di una singola organizzazione. La gestione, l'operazione, la proprietà, la dislocazione della Cloud privata tuttavia può essere anche indipendente dall'organizzazione che la usa.
- **Cloud di comunità (Community Cloud):**
  - L'infrastruttura è disponibile ad una comunità di organizzazioni che hanno uno scopo comune (ad esempio missione, requisiti di sicurezza, conformità a regole comuni, etc.)
- **Cloud pubblica:**
  - L'infrastruttura è disponibile in generale al pubblico. La gestione può essere pubblica o privata. La dislocazione è presso il fornitore di servizi.
- **Cloud ibrida:**
  - L'infrastruttura è una combinazione di due o più infrastrutture Cloud (private, di comunità o pubbliche) che sono collegate in modo da garantire forme di portabilità ad esempio di dati o applicazioni.

Isolamento

- I modelli di **isolamento** nel Cloud (spesso ignorati) sono importanti e si dividono in:
  - Infrastrutture dedicate
  - Infrastrutture "multi-tenant" (con diversi [tipi di] clienti)
- Il tipo di isolamento è importante per molti aspetti, come:
  - Segmentazione delle risorse
  - Protezione dei dati
  - Sicurezza delle applicazioni
  - Auditing
  - Disaster recovery

Che differenza c'è…

- … tra virtualizzazione e Cloud computing?

- Risposta:

Virtualizzazione?

- Il Cloud computing può anche essere fornito *senza* l'utilizzo di tecnologie di virtualizzazione.
  - Spesso tuttavia l'utilizzo di tecnologie di virtualizzazione consente di ridurre i costi operativi e in conto capitale.
  - Essere in grado di fornire molto rapidamente delle macchine virtuali non è comunque efficiente, se servono diversi mesi per effettuare il provisioning e l'installazione degli host fisici.
    - Inoltre, il tempo impiegato per la fornitura dello strato di virtualizzazione è recuperato dai risparmi associati al non dover utilizzare server fisici?
    - Importanza di avere tool di installazione, monitoring e accounting il più possibile automatizzati.
- Ma che cosa si intende con *virtualizzazione*?

Riassumendo: virtualizzazione vs. Cloud computing

- **Installazione/reinstallazione** di server o di applicazioni su VM di per sé ***non è Cloud computing***.
- Verificare con le **5 caratteristiche del Cloud** mostrate precedentemente:
  - Self-service, on-demand → **NO** (tipicamente è un dipartimento IT che fornisce le VM)
  - Accesso attraverso la rete → **NO** (deployment limitato a "internal customers")
  - Pool di risorse → **SÌ**
  - Elasticità → **NO** (tipicamente è un dipartimento IT che deve installare sistema operativo + software, e non necessariamente in modo scalabile)
  - Pagamento a consumo → **NO** (spesso il billing non viene fatto a consumo ma in modo tradizionale)
- **Un esempio di virtualizzazione che non è Cloud?** (ma che è complementare al Cloud)

Cloud Computing

Part 1: What is Cloud Computing?

University of Padova
Department of Mathematics

July 14, 2014

Summary

Computing utilites - Vision

- 1969, Leonard Kleinrock one of the chief scientists of the original ARPANET:

"As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country."

Referred to as utility computing or, recently (since 2007), as cloud computing:

- users access services based on their requirements without regard to where the services are hosted
- denotes the infrastructure as a "cloud" from which businesses and users can access applications as services from anywhere in the world and on demand
- cloud computing can be classified as a new paradigm for the dynamic provisioning of computing services supported by state-of-the-art data centers employing virtualization technologies for consolidation and effective utilization of resources.

Computing utilites - Enablers

- One of the most diffuse views of cloud computing:

"I don't care where my servers are, who manages them, where my documents are stored, or where my applications are hosted. I just want them always available and access them from any device connected through Internet. And I am willing to pay for this service for as a long as I need it."

- strong similarities to the way we use other services, such as water and electricity
- turns IT services into utilities
- made possible by the effective composition of several technologies, which have reached the appropriate maturity level:
  - Web 2.0 technologies ⇒ Internet into a rich application and service delivery platform
  - Service orientation ⇒ to deliver capabilities with familiar abstractions
  - Virtualization ⇒ confers on cloud computing the necessary degree of customization, control, and flexibility for building production and enterprise systems.

Cloud - The term

The term *cloud* has historically been used in the telecommunications industry:

- abstraction of e network in system diagrams
- became the symbol of the most popular computer network: the Internet

This meaning also applies to cloud computing, which refers to an Internet-centric way of computing. Internet plays a fundamental role.

"Cloud computing refers to both the applications delivered as services over the Internet and the hardware and system software in the datacenters that provide those services."

Touching on the entire stack:

- XaaS ⇒ everything as a service
- different component of a system:
  - delivered
  - measured
  - priced, as a service

Cloud Computing at a glance

The utility-oriented nature of cloud computing:

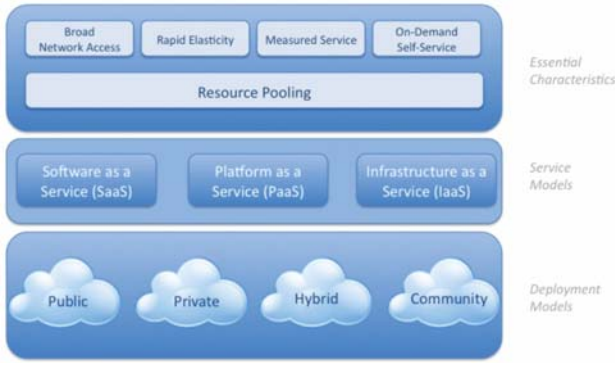
"A cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers."

Buyya et.al

Cloud Computing - NIST Reference Model

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Composed of: 5 essential characteristics, 3 service models, 4 deployment models.



Cloud Computing - 5 essential characteristics

- On-demand self-service
  - a consumer can unilaterally provision computing capabilities
  - doesn't require human interaction with each service provider
- Broad network access
  - capabilities available over the network
  - accessed through standard mechanisms
  - heterogeneous, thin or thick client platforms
- Resource pooling
  - provider's computing resources are pooled to serve multiple consumers
  - multi-tenant model
  - different physical and virtual resources dynamically assigned and reassigned according to consumer demand
  - customer generally has no control or knowledge over the exact location
  - may be able to specify location at a higher level of abstraction

Cloud Computing - 5 essential characteristics

...continued

- Rapid elasticity
  - elastic provision of capabilities, commensurate with demand
  - to the consumer, appear to be unlimited and can be appropriated in any quantity at any time
- Measured service
  - automatic control and optimization of resource
  - metering capability appropriate to the type of service
  - resource usage can be monitored, controlled, and reported
  - transparency for both the provider and consumer of the utilized service

Cloud Computing - 3 service models

- Software as a Service (SaaS)
  - consumers use the provider's applications running on a cloud infrastructure
  - applications accessible from various client devices
  - possible exception of limited user-specific application configuration settings
- Platform as a Service (PaaS)
  - consumers deploy onto the cloud infrastructure
  - consumers create or acquire applications using tools supported by the provider
  - consumers has control over the deployed applications and possibly configuration settings for the application-hosting environment
- Infrastructure as a Service (IaaS)
  - consumers provision computing resources (e.g., processing, storage, network)
  - consumers are able to deploy and run arbitrary software, including OS and apps
  - consumers have control over OS, storage, and deployed applications
  - possibly limited control of select networking components (e.g., host firewalls)

Consumers does not manage or control the underlying cloud infrastructure !!!

Cloud Computing - 4 deployment models

- Private cloud
  - cloud infrastructure provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units)
  - may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises
- Community cloud
  - cloud infrastructure provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy , and compliance considerations)
  - may be owned, managed , and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises

Cloud Computing - 4 deployment models

...continued

- Public cloud
  - cloud infrastructure provisioned for open use by the general public
  - may be owned, managed, and operated by a business, academic, or government organization, or some combination of them
  - it exists on the premises of the cloud provider
- Hybrid cloud
  - cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public)
  - remain unique entities , but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)

CS5412: ANATOMY OF A CLOUD

Lecture VII

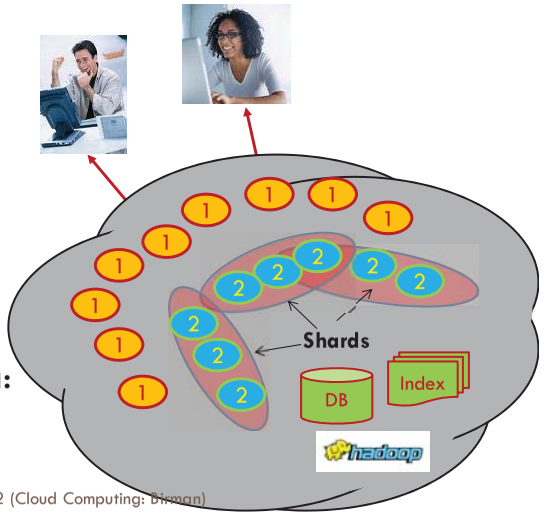
Ken Birman

How are cloud structured?

- Clients talk to clouds using web browsers or the web services standards
  - But this only gets us to the outer "skin" of the cloud data center, not the interior
  - Consider Amazon: it can host entire company web sites (like Target.com or Netflix.com), data (AC3), servers (EC2) and even user-provided virtual machines!

Big picture overview

- Client requests are handled in the "first tier" by
  - PHP or ASP pages
  - Associated logic
- These lightweight services are fast and very nimble
- Much use of caching: the second tier



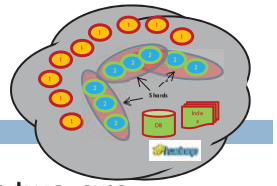
Many styles of system

- Near the edge of the cloud, focus is on vast numbers of clients and rapid response
- Inside we find high volume services that operate in a pipelined manner, asynchronously
- Deep inside the cloud we see a world of virtual computer clusters that are scheduled to share resources and on which applications like MapReduce (Hadoop) are very popular

In the outer tiers replication is key

- We need to replicate
  - Processing: each client has what seems to be a private, dedicated server (for a little while)
  - Data: as much as possible, that server has copies of the data it needs to respond to client requests without any delay at all
  - Control information: the entire structure is managed in an agreed-upon way by a decentralized cloud management infrastructure

What about the "shards"?



- The caching components running in tier two are central to the responsiveness of tier-one services
  - The basic idea is to always used cached data if at all possible, so the inner services (here, a database and a search index stored in a set of files) are shielded from "online" load
  - We need to replicate data within our cache to spread loads and provide fault-tolerance
  - But not everything needs to be "fully" replicated. Hence we often use "shards" with just a few replicas

Sharding used in many ways

- The second tier could be any of a number of caching services:
  - Memcached: a sharable in-memory key-value store
  - Other kinds of DHTs that use key-value APIs
  - Dynamo: A service created by Amazon as a scalable way to represent the shopping cart and similar data
  - BigTable: A very elaborate key-value store created by Google and used not just in tier-two but throughout their "GooglePlex" for sharing information
- Notion of sharding is cross-cutting
  - Most of these systems replicate data to some degree

Do we *always* need to shard data?

- Imagine a tier-one service running on 100k nodes
  - Can it ever make sense to replicate data on the entire set?
- Yes, if some kinds of information might be so valuable that almost every external request touches it
  - Must think hard about patterns of data access and use
  - Some information needs to be heavily replicated to offer blindingly fast access on vast numbers of nodes
  - The principle is similar to the way Beehive operates.
    - Even if we don't make a dynamic decision about the level of replication required, the principle is similar
    - We want the level of replication to match level of load and the degree to which the data is needed on the critical path

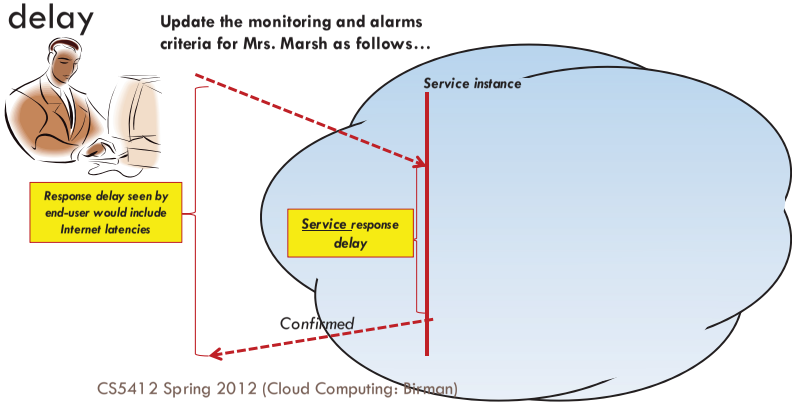
And it isn't just about updates

- Should also be thinking about patterns that arise when doing reads ("queries")
  - Some can just be performed by a single representative of a service
  - But others might need the parallelism of having several (or even a huge number) of machines do parts of the work concurrently
- The term sharding is used for data, but here we might talk about "parallel computation on a shard"

What does "critical path" mean?

10

- Focus on delay until a client receives a reply
- Critical path is formed by actions that contribute to this delay



What if a request triggers updates?

11

- If the updates are done "asynchronously" we might not experience much delay on the critical path
  - Cloud systems often work this way
  - Avoids waiting for slow services to process the updates but may force the tier-one service to "guess" the outcome
  - For example, could optimistically apply update to value from a cache and just hope this was the right answer
- Many cloud systems use these sorts of "tricks" to speed up response time

First-tier parallelism

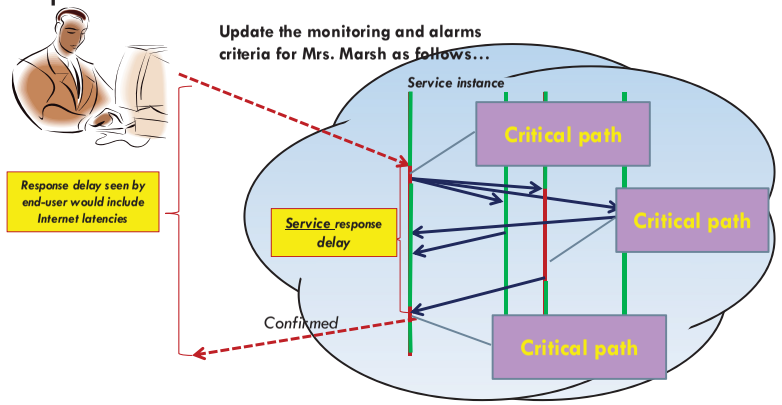
12

- Parallelism is vital to speeding up first-tier services
- Key question:
  - Request has reached some service instance X
  - Will it be faster…
    - … For X to just compute the response
    - … Or for X to subdivide the work by asking subservices to do parts of the job?
- Glimpse of an answer
  - Werner Vogels, CTO at Amazon, commented in one talk that many Amazon pages have content from 50 or more parallel subservices that ran, in real-time, on your request!

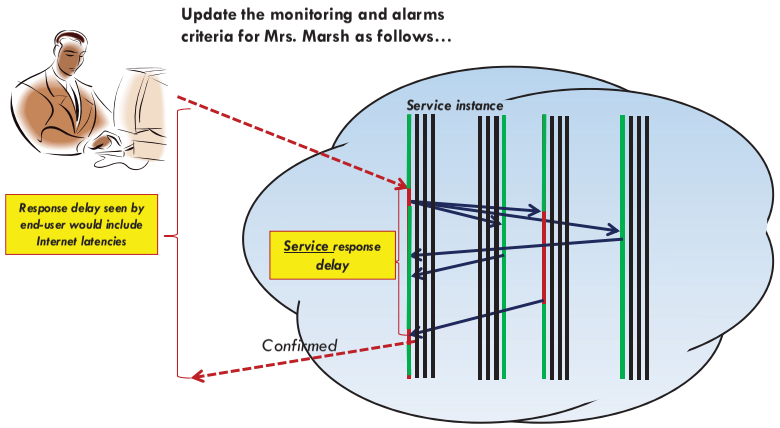
What does "critical path" mean?

13

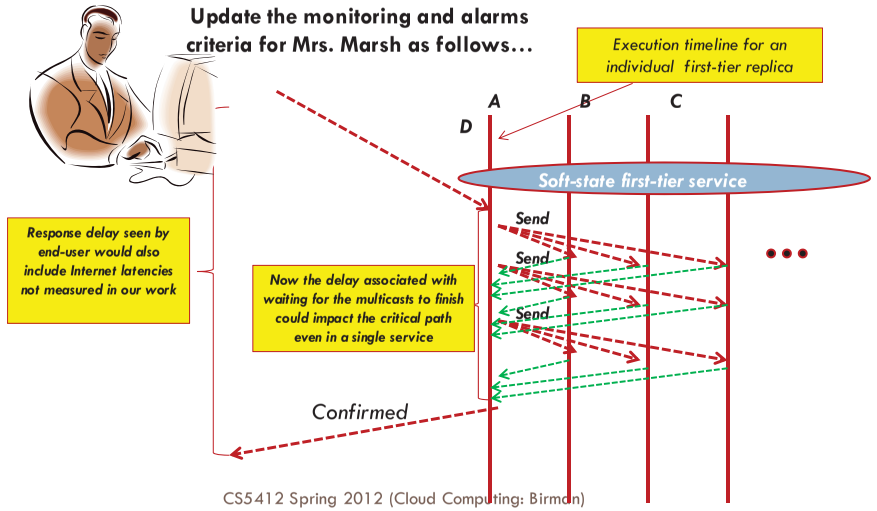
- In this example of a parallel read-only request, the critical path centers on the middle "subservice"



With replicas we just load balance



But when we add updates….



What about sending updates without waiting?

- Several issues now arise
  - Are all the replicas applying updates in the same order?
    - Might not matter unless the same data item is being changed
    - But then clearly we do need some "agreement" on order
  - What if the leader replies to the end user but then crashes and it turns out that the updates were lost in the network?
    - Data center networks are surprisingly lossy at times
    - Also, bursts of updates can queue up
- Such issues result in *inconsistency*

Eric Brewer's CAP theorem

- In a famous 2000 keynote talk at ACM PODC, Eric Brewer proposed that "you can have just two from Consistency, Availability and Partition Tolerance"
  - He argues that data centers need very snappy response, hence availability is paramount
  - And they should be responsive even if a transient fault makes it hard to reach some service. So they should use cached data to respond faster even if the cached entry can't be validated and might be stale!
- Conclusion: weaken consistency for faster response

CAP theorem

- A proof of CAP was later introduced by MIT's Seth Gilbert and Nancy Lynch
  - Suppose a data center service is active in two parts of the country with a wide-area Internet link between them
  - We temporarily cut the link ("partitioning" the network)
  - And present the service with conflicting requests
- The replicas can't talk to each other so can't sense the conflict
- If they respond at this point, inconsistency arises

Is inconsistency a bad thing?

- How much consistency is really needed in the first tier of the cloud?
  - Think about YouTube videos. Would consistency be an issue here?
  - What about the Amazon "number of units available" counters. Will people notice if those are a bit off?
- Puzzle: can you come up with a general policy for knowing how much consistency a given thing needs?

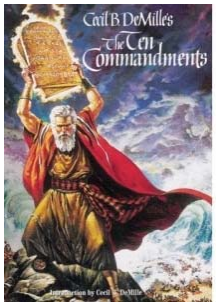
THE WISDOM OF THE SAGES

eBay's Five Commandments

- As described by Randy Shoup at LADIS 2008

Thou shalt…

1. **Partition Everything**
2. **Use Asynchrony Everywhere**
3. **Automate Everything**
4. **Remember: Everything Fails**
5. **Embrace Inconsistency**



Vogels at the Helm

- Werner Vogels, CTO at Amazon.com …
- He was involved in building a new shopping cart service
  - The old one used strong consistency for replicated data
  - New version was build over a DHT, like Chord, and has weak consistency with eventual convergence
- This weakens guarantees … but
  - ***Speed matters more than correctness***

James Hamilton's advice

- Key to scalability is decoupling, loosest possible synchronization
- *Any* synchronized mechanism is a risk
  - His approach: create a committee
  - Anyone who wants to deploy a highly consistent mechanism needs committee approval

…. They don't meet very often

Consistency

Consistency technologies just don't scale!

But inconsistency brings risks too!

My rent check bounced? That can't be right!

- Inconsistency causes bugs
  - Clients would never be able to trust servers… a free-for-all

- Weak or "best effort" consistency?
  - Strong security guarantees demand consistency
  - Would you trust a medical electronic-health records system or a bank that used "weak consistency" for better scalability?

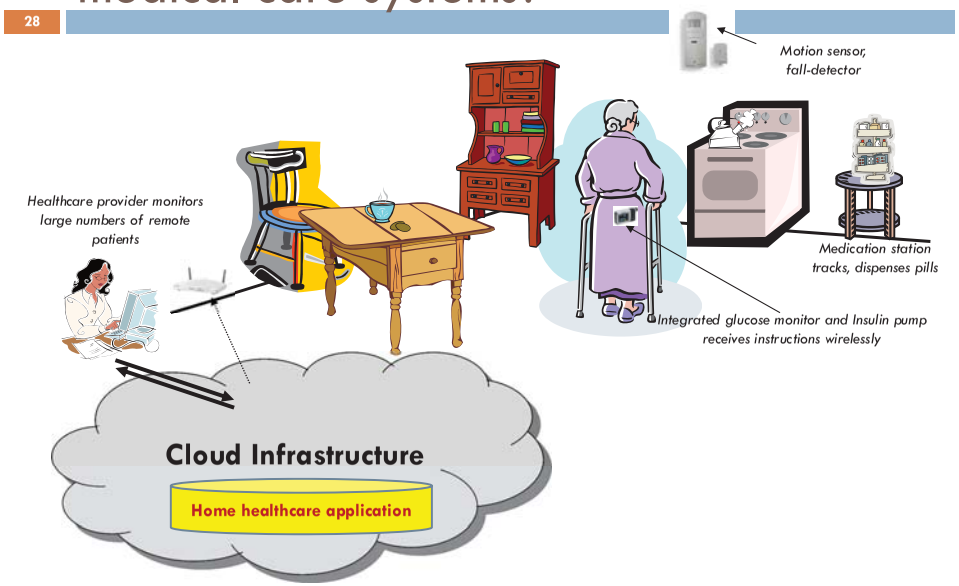
Puzzle: Is CAP valid in the cloud?

- Facts: data center networks don't normally experience partitioning failures
  - Wide-area links do fail
  - But most services are designed to do updates in a single place and mirror read-only data at others
  - So the CAP scenario used in the proof can't arise
- Brewer's argument about not waiting for a slow service to respond does make sense
  - Argues for using any single replica you can find
  - But does this preclude that replica being consistent?

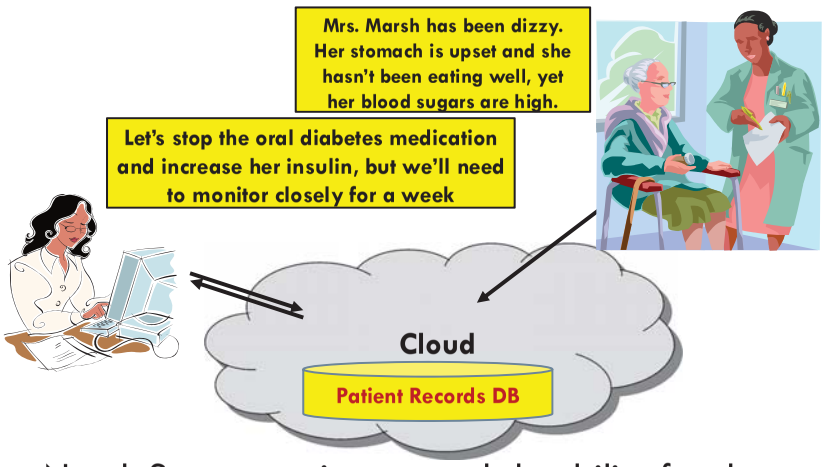
What does "consistency" mean?

- We need to pin this basic issue down!
- As used in CAP, consistency is about two things
  - First, that updates to the same data item are applied in some agreed-upon order
  - Second, that once an update is acknowledged to an external user, it won't be forgotten
- Not all systems need both properties

What properties are needed in remote medical care systems?

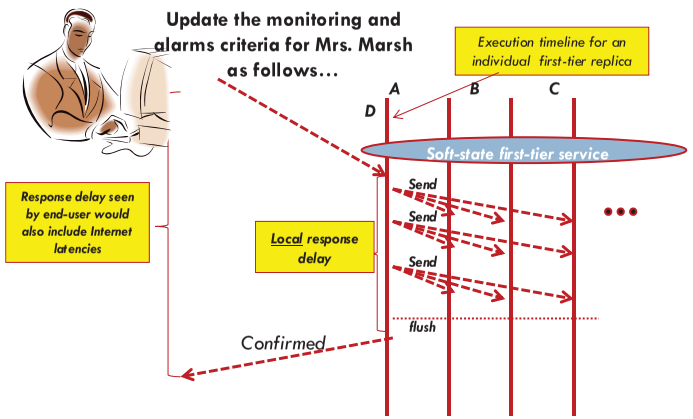


Which matters more: fast response, or durability of the data being updated?



- Need: Strong consistency and durability for data

What if we were doing online monitoring?



- An online monitoring system might focus on real-time response and value consistency, yet be less concerned with durability

Why does monitoring have weaker needs?

- When a monitoring system goes "offline" the device turns a red light or something on
  - Later, on recovery, the monitoring policy may have changed and a node would need to reload it
  - Moreover, with in-memory replication we may have a strong enough guarantee for most purposes
- Thus if durability costs enough to slow us down, we might opt for a weaker form of durability in order to gain better scalability and faster responses!

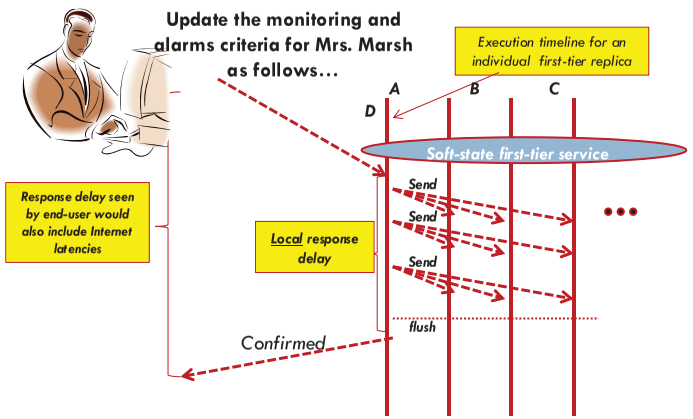
This illustrates a challenge!

- Cloud systems just can't be approached in a one-size fits all manner
- For performance-intensive scalability scenarios we need to look closely at tradeoffs
  - Cost of stronger guarantee, versus
  - Cost of being faster but offering weaker guarantee
- If systems builders blindly opt for strong properties when not needed, we just incur other costs!
  - Amazon: Each 100ms delay reduces sales by 1%!

Properties we might want

- Consistency: Updates in an agreed order
- Durability: Once accepted, won't be forgotten
- Real-time responsiveness: Replies with bounded delay
- Security: Only permit authorized actions by authenticated parties
- Privacy: Won't disclose personal data
- Resilience: Failures can't prevent the system from providing desired services
- Coordination: actions won't interfere with one another

Fast response with consistency



This mixture of features gives us consistency, an in-memory replication guarantee ("amnesia freedom"), but not full durability

Does CAP apply deeper in the cloud?

- The principle of wanting speed and scalability certainly is universal
- But many cloud services have strong consistency guarantees that we take for granted but depend on
- Marvin Theimer at Amazon explains:
  - Avoid costly guarantees that aren't even needed
  - But sometimes you just need to guarantee something
  - Then, be clever and engineer it to scale
  - And expect to revisit it each time you scale out 10x

Cloud services and their properties

Service	Properties it guarantees
Memcached	No special guarantees
Google's GFS	File is current if locking is used
BigTable	Shared key-value store with many consistency properties
Dynamo	Amazon's shopping cart: eventual consistency
Databases	Snapshot isolation with log-based mirroring (a fancy form of the ACID guarantees)
MapReduce	Uses a "functional" computing model within which offers very strong guarantees
Zookeeper	Yahoo! file system with sophisticated properties
PNUTS	Yahoo! database system, sharded data, spectrum of consistency options
Chubby	Locking service… very strong guarantees

Is there a conclusion to draw?

- One thing to notice about those services…
  - Most of them cost 10's or 100's of millions to create!
  - Huge investment required to build strongly consistent and scalable and high performance solutions
  - Oracle's current parallel database: billions invested
- CAP isn't about telling Oracle how to build a database product…
  - CAP is a warning to you that strong properties can easily lead to slow services
  - But thinking in terms of weak properties is often a successful strategy that yields a good solution and requires less effort

Core problem?

- When can we safely sweep consistency under the rug?
  - If we weaken a property in a safety critical context, something bad can happen!
  - Amazon and eBay do well with weak guarantees because many applications just didn't need strong guarantees to start with!
  - By embracing their weaker nature, we reduce synchronization and so get better response behavior
- But what happens when a wave of high assurance applications starts to transition to cloud-based models?

GeoServer nel Cloud

Un caso di studio sulle modifiche architetturali nel passaggio a piattaforme Cloud

Federico Cacco

Laurea Magistrale in Informatica

Università degli Studi di Padova
Dipartimento di Matematica

11 Ottobre 2013

Sommario

Scopo del progetto

Le architetture delle piattaforme Cloud sono del tutto diverse da quelle convenzionali

- Applicazioni ⇒ Fornite come servizi web
- Istanza ⇒ Entità operativa che fruisce i servizi
- Elasticità ⇒ Capacità di adattarsi (scaling) all'esigenza corrente
  - Ottenuta mediante la replicazione delle istanze

Applicazioni progettate per piattaforme convenzionali sono inadeguate agli ambienti Cloud

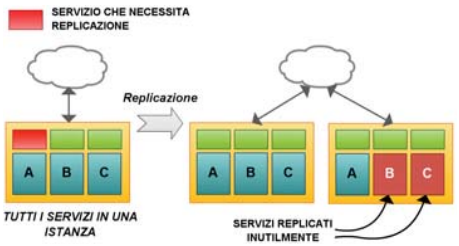
Quali considerazioni architetturali sono necessarie?

Scomposizione dei servizi

Necessario replicare solamente i componenti che ne hanno reale necessità

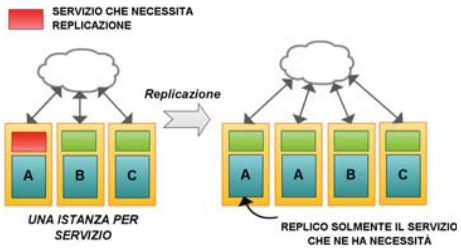
Tutti i servizi in una unica istanza

- *Per scalare un servizio devo replicare l'istanza che fornisce tutti i servizi*



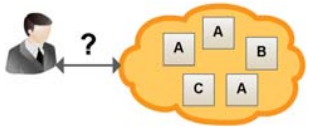
Una servizio per istanza

- *Per scalare un servizio posso replicare solamente l'istanza che lo fornisce*



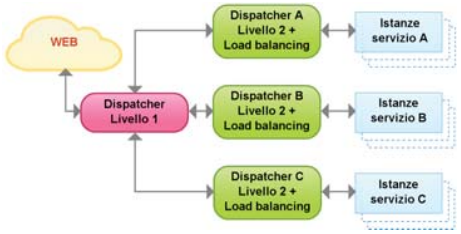
Rete di Dispatcher

La replicazione delle istanze che compongono il web server deve essere trasparente



- L'utente non deve preoccuparsi di inviare la richiesta all'istanza in quel momento libera
- Deve essere presente un unico punto d'accesso al web server
  - *Indipendentemente dal numero di replicazioni e dal numero di servizi forniti*

Soluzione: Rete di dispatcher



- Secondo livello:
  - *Gestisce replicazioni di istanze che forniscono lo stesso servizio*
  - *Bilancia il carico di lavoro*
- Primo livello
  - *Instrada le chiamate al corretto dispatcher di secondo livello*
  - *Espone un unico punto d'accesso al web server*

Configurazione comune delle istanze

- Tutte le istanze devono avere la medesima configurazione
- Modifiche alla configurazione di una istanza devono ripercuotersi su tutte le altre

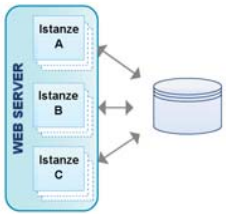
Soluzione 1: Configurazione in uno spazio condiviso

Soluzione 2: Configurazione locale sincronizzata

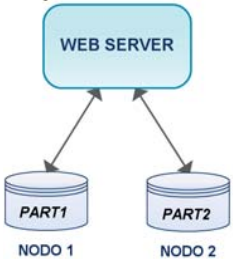
La scelta dipende dall'architettura dei servizi e dalla piattaforma Cloud utilizzata

Memorizzazione dei dati

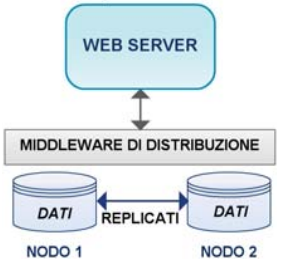
- Database soggetto a numerosi accessi simultanei
- Necessità di una sua distribuzione
  - *Partizionamento*
  - *Replicazione*



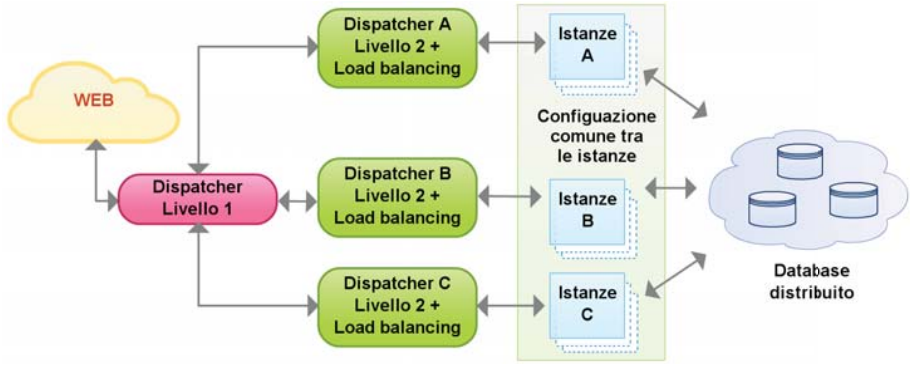
Distribuzione mediante partizionamento



Distribuzione mediante replicazione



Visione d'insieme



GeoServer

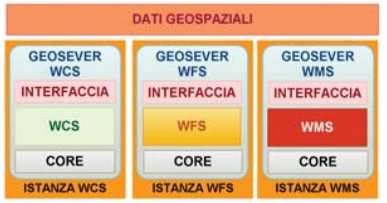
Caso di studio concreto: GeoServer

Server web geospaziale che fornisce i servizi

- *Web Coverage Service*
- *Web Feature Service*
- *Web Map Service*

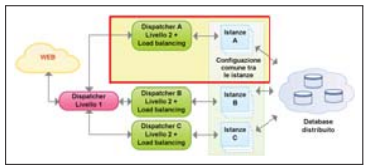
Servizi con differenti carichi computazionali

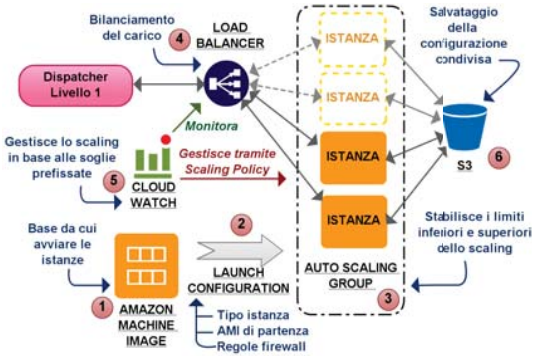
- *Scomposizione per poter avviare istanze contenenti i singoli servizi*



Strumenti AWS utilizzati

Ambiente Cloud utilizzato: Amazon Web Services

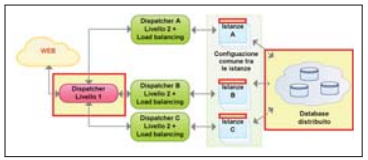




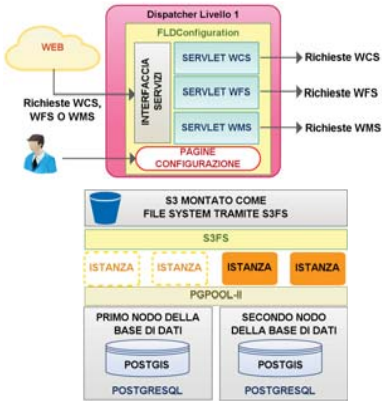
Metrica per lo scaling ⇒ Latenza

- *Rilevata nei Load Balancer di secondo livello*

Strumenti esterni ad AWS utilizzati



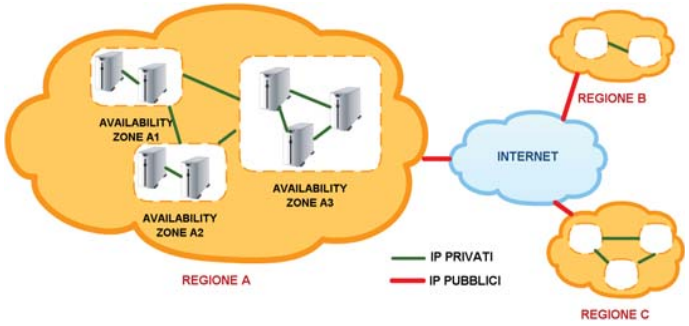
- Dispatcher di primo livello



- S3FS (Middleware supporto S3)
- Pgpool-II (Middleware di distribuzione)
- PostgreSQL (Data Base Management System)
- PostGIS (Estensione per dati geospaziali)

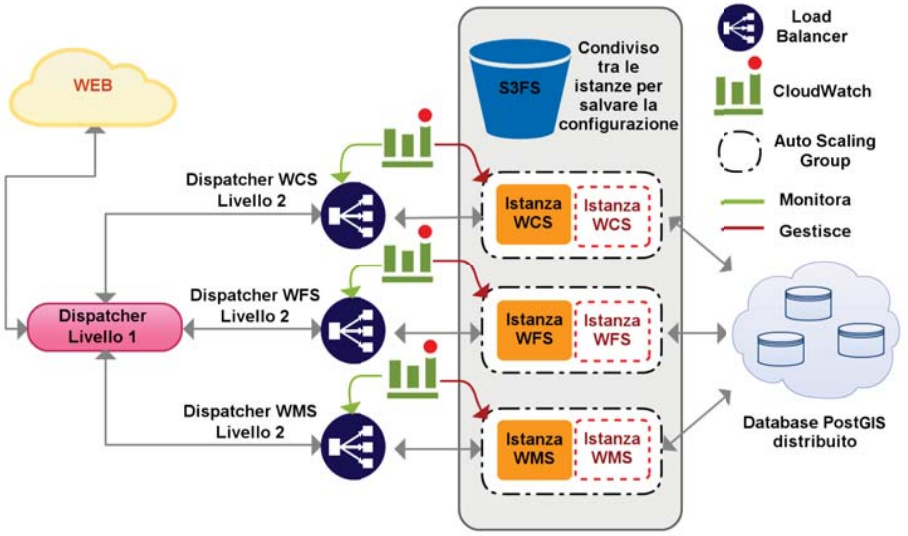
Regioni e Availability Zone

- Regione: Sedi su cui è collocato AWS nel mondo
- Availability Zone: Sotto suddivisione delle Regioni



Le Availability Zone consentono di implementare architetture affidabili impedendo la diffusione dei guasti

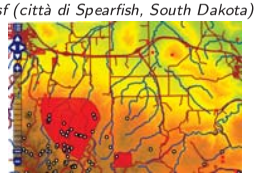
Visione d'insieme



Configurazione dei test

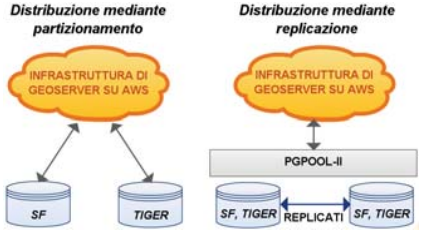
- Simulati utenti contemporanei che invocano i servizi
- 2 serie di dati geospaziali suddivisi in 2 workspace

sf (città di Spearfish, South Dakota)



tiger (città di New York)

- 2 configurazioni di distribuzione della base di dati
  - *Partizionamento*
  - *Replicazione*



Availability Zone

La distribuzione nelle Availability Zone per aumentare l'affidabilità introduce latenze?

- Simulati 5 utenti contemporanei

Richiesta	Diversa AZ		Stessa AZ	
	Lat. media (ms)	Throughput (rich/sec)	Lat. media (ms)	Throughput (rich/sec)
WMS_getMap (sf)	606	1,6180	625	1,6662
WMS_getMap (tiger)	580	1,7667	586	1,6785
WMS_getMap_multilayer_3_svg (sf)	943	1,7620	1001	1,6722
WMS_getMap_multilayer_3_svg (tiger)	2412	1,7518	2610	1,6360
Totale	1134	6,9874	1205	6,5174

Distribuire l'applicazione tra più Availability Zone non introduce latenze

Partizionato VS Replicato - 2 workspace

La distribuzione mediante partizionamento è più efficiente nel caso di richieste riferite a workspace diverse?

- Simulati 16 utenti contemporanei

Richiesta	DB partizionato		DB replicato	
	Lat. media (ms)	Throughput (rich/sec)	Lat. media (ms)	Throughput (rich/sec)
WMS_getMap (sf)	524	3,4943	818	2,7626
WMS_getMap (tiger)	577	3,4895	807	2,7604
WMS_getMap_multilayer_3_svg (sf)	975	3,4847	1322	2,7477
WMS_getMap_multilayer_3_svg (tiger)	2516	3,4379	2870	2,7571
Totale	1146	13,7263	1452	10,8429

Con chiamate relative a dati in differenti workspace la distribuzione mediante partizionamento risulta più efficiente

Partizionato VS Replicato - 1 workspace

La distribuzione mediante replicazione è più efficiente nel caso di richieste riferite sempre alla stessa workspace?

- Simulati 16 utenti contemporanei

Richiesta	DB partizionato		DB replicato	
	Lat. media (ms)	Throughput (rich/sec)	Lat. media (ms)	Throughput (rich/sec)
WMS_getMap (tiger)	634	3,2172	1021	2,6029
WMS_getMap_multilayer_3_svg (tiger)	2972	3,1615	3227	2,5524
WMS_getMap_multilayer_cql (tiger)	684	3,2058	925	2,5952
WMS_getMap_multilayer_filter (tiger)	693	3,1974	989	2,5944
Totale	1248	12,5958	1542	10,1734

Distribuzione mediante partizionamento nuovamente più efficiente

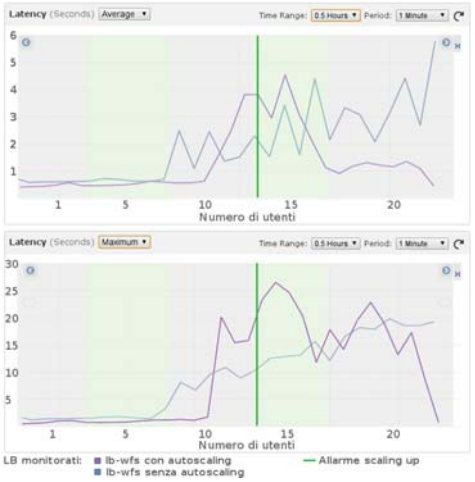
Per piccole distribuzioni il middleware non introduce benefici

Test sullo scaling up

Lo scaling up riduce la latenza quando essa diventa eccessiva?

Soglie scaling:

- Scaling up: lat. $>$ 15 s (MAX)



Latenza ridotta dallo scaling up

Ritardo tra allarme ed effettiva diminuzione della latenza

Test sullo scaling down

Vengono liberate le risorse quando non sono più necessarie?

Soglie scaling:

- Scaling up: lat. $>$ 15 s (MAX)
- Scaling down: lat. $<$ 1 s (MED)

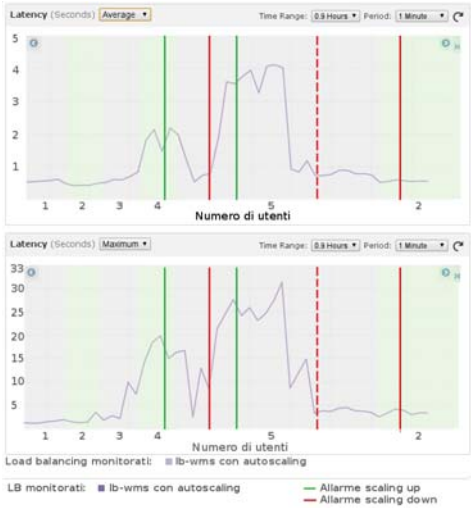
Tempo attivazione istanza $>$ Tempo disattivazione istanza

Test sullo scaling down - oscillazione

Importante utilizzare soglie di scaling corrette!

Soglie scaling:

- Scaling up: lat. $>$ 10 s (MAX)
- Scaling down 1: lat. $<$ 1 s (MED)
- Scaling down 2: lat. $<$ 0.6 s (MED)



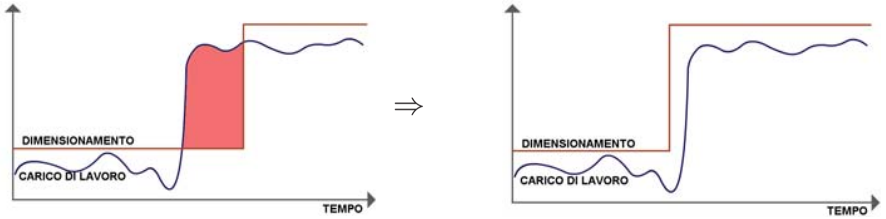
Oscillazione dovuta ad una soglia di scaling down troppo alta

Conclusioni

- La latenza si rivela una buona metrica per determinare se il dimensionamento è coerente con il carico di lavoro
- Rendere l'architettura più affidabile mediante una sua distribuzione su più Availability Zone non ne pregiudica le prestazioni
- Nel caso di richieste fatte a più workspace, la distribuzione mediante partizionamento si rileva più efficiente di quella realizzata mediante replicazione
- Lo scaling up è una operazione molto più onerosa, in termini di tempo e risorse, dello scaling down
- Difficile determinare soglie di scaling corrette

Scomposizione servizi

- Predire i picchi di carico in modo da anticipare lo scaling dei servizi



- Testare la distribuzione mediante replicazione con scaling up di dimensioni maggiori