Analisi e rappresentazione del comportamento dei ritardi di rete dei principali fornitori di Cloud Computing

Candidato
Alberto Bagnacani

Relatore
Prof. Paolo Bellavista

Correlatore
Dott. Ing. Luca Foschini

Anno accademico 2017-2018

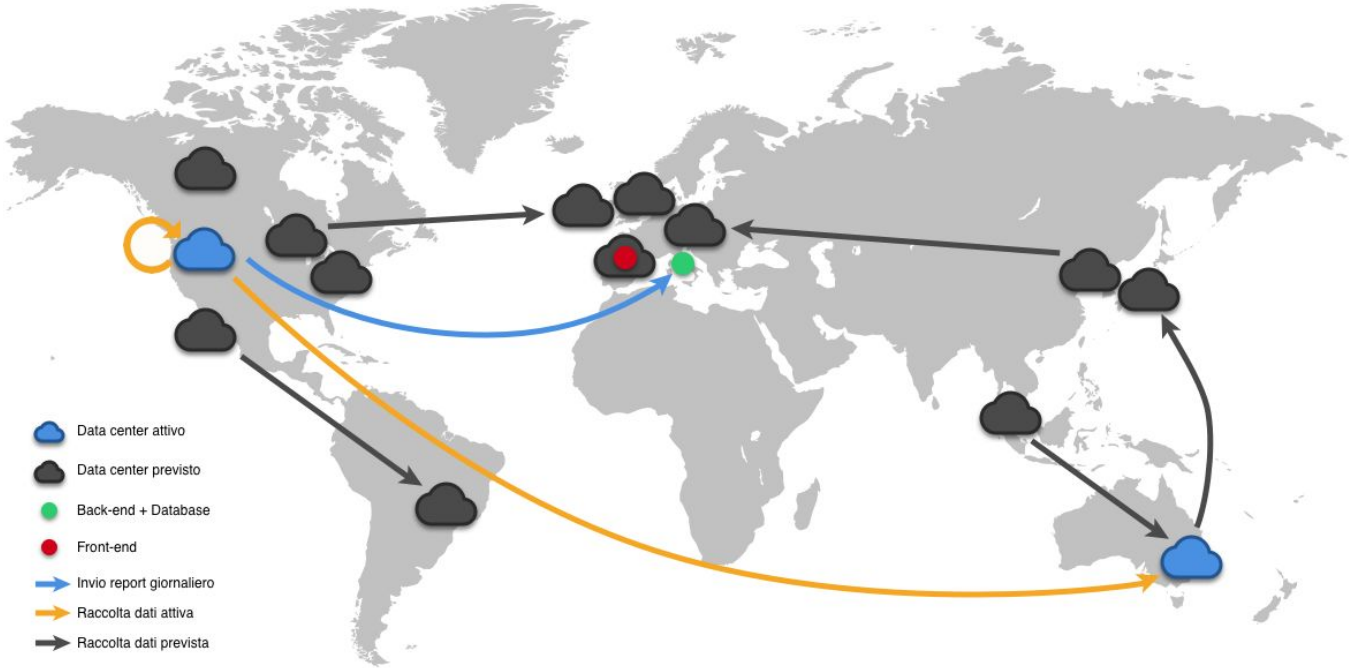
Introduzione

Cloud Computing paradigma in **continua crescita** → Necessità di potersi **basare** su un **modello** affidabile di **ritardi** all'interno di differenti aree terrestri.

Esigenza di uno strumento per il **monitoraggio** della **latenza** tra datacenter → Progetto full-stack per la **raccolta**, **analisi** e **presentazione** della **latenza** di comunicazione che caratterizza i vari **data center,** per diversi Cloud Provider.

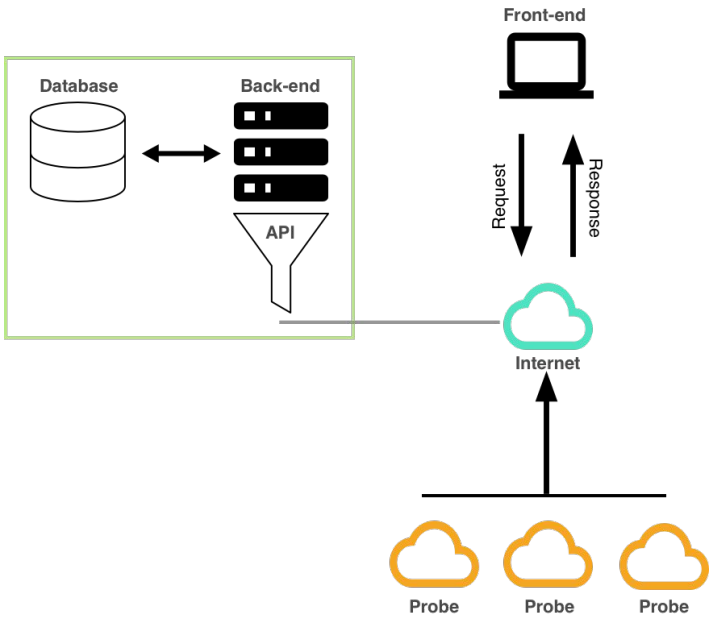
Cloud Provider **esaminati**: Amazon Web Services, Google Cloud Platform (Microsoft Azure, IBM Cloud)

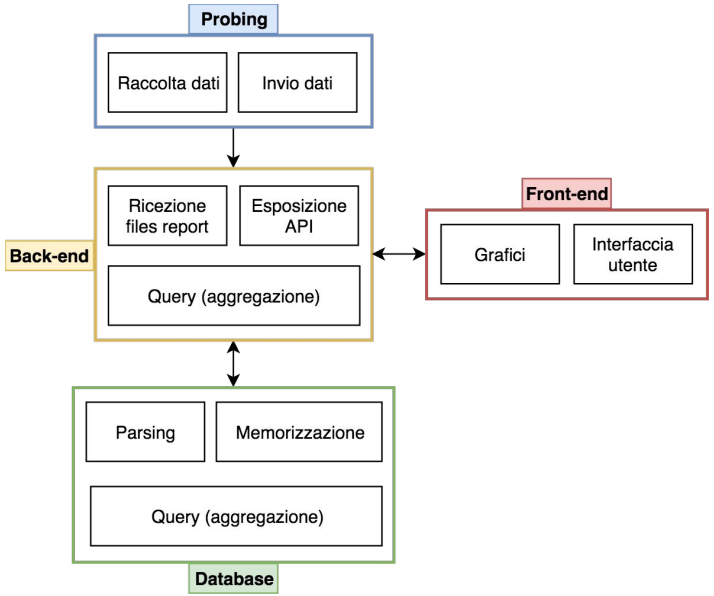
Obiettivo e deployment



- Sistema di **monitoraggio dinamico unificato** → Progettazione del deployment
- **Deployment world wide** a regime dell'algoritmo di raccolta dati

Architettura e funzionalità





Tecnologie

MongoDB

Database
NoSQL

Expedia
Bosch
Cisco

Express

Framework
back-end

IBM
Accenture
Uber

React

Framework
front-end

Facebook
AirBnb
Bloomberg

Node.js

Ambiente
runtime
JavaScript

Google
Intel
NASA

Implementazione

Probing

```
ping $DST_HOST |
#...
while read LINE; do
        printf "$PROVIDER, $SRC_ZONE, $DST_ZONE, $SRC_HOST, $DST_HOST, $ICMP_SEQ, $TTL, $TIME, $TIMESTAMP \n" >> $FILE
done

curl -F “data=@$FILE” $SERVER # invio eseguito tramite strumento “cron”
```

- Algoritmo di **raccolta** ed invio dati realizzato in linguaggio **shell**
- Files di report giornalieri in formato CSV
- Soluzione semplice ed efficace

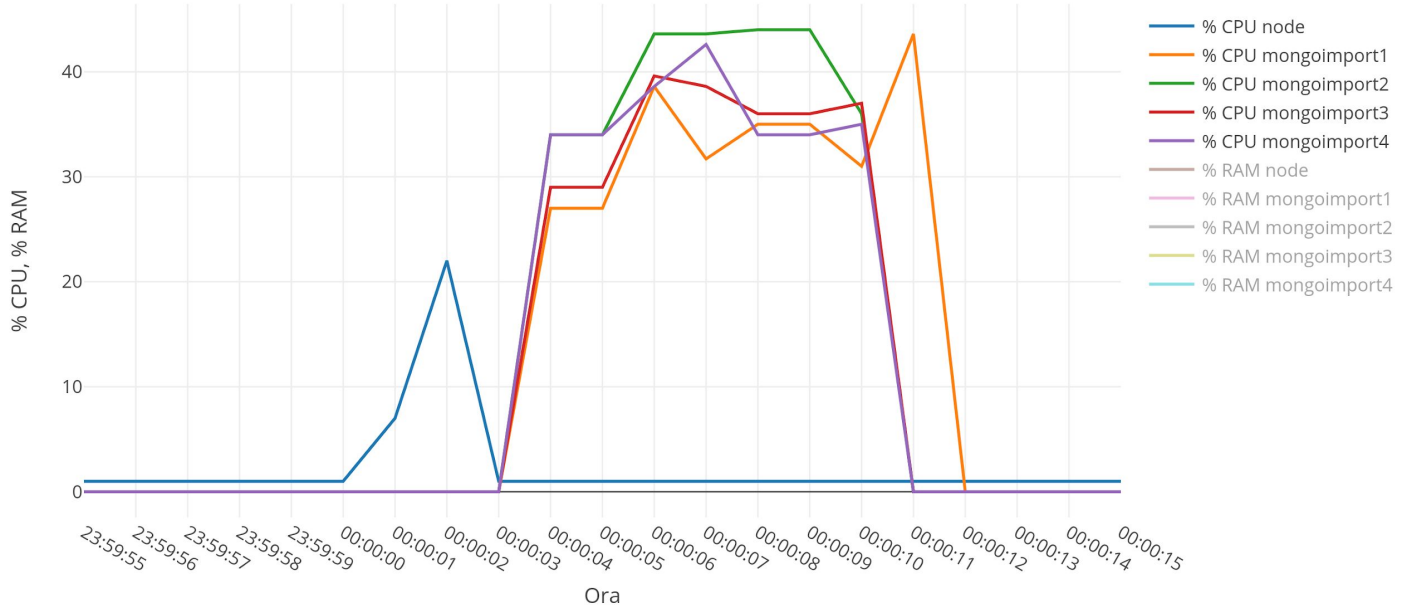
Implementazione

Persistenza

```
mongoimport -h IP -d database -c collection
-u albertobagnacani -p password
--type csv --columnsHaveTypes
--fields "provider.string\(\), src_zone.string\(\), dst_zone.string\(\), src_host.string\(\), dst_host.string\(\), icmp_seq.int32\(\), ttl.int32\(\),
time.double\(\), timestamp.date\(\)"'
```

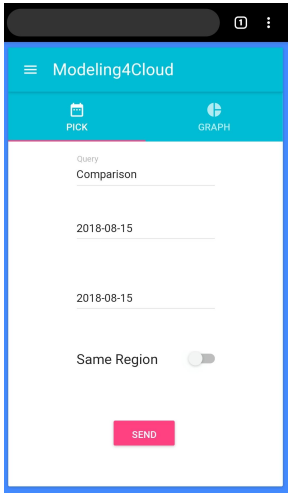
- **Mongoimport** strumento per **parsing** e **memorizzazione** dei files su database
- Definizione dei tipi dei campi
- Buona efficienza

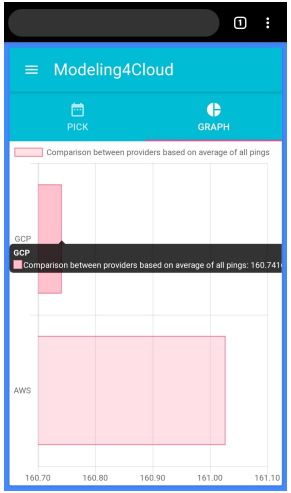
Analisi software



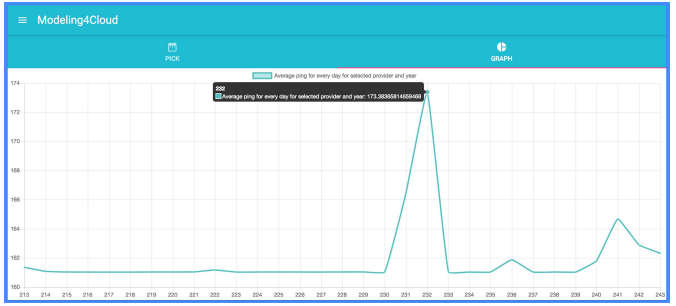
- Un processo “mongoimport” per ogni probe
- Analisi delle caratteristiche del software: evidenziazione della **scalabilità** del sistema

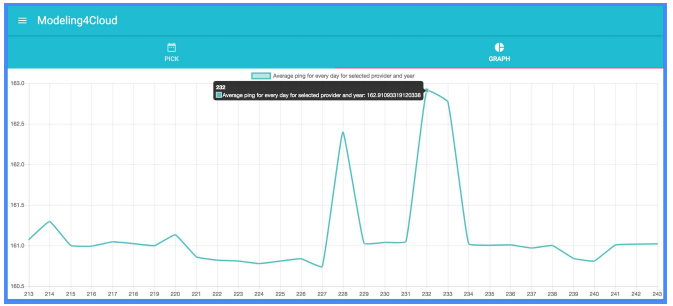
Risultati





AWS - Agosto 2018





GCP - Agosto 2018

- Interfaccia grafica semplice e reattiva
- **Immediatezza** del risultato grazie ai grafici
- **Stima** visiva della **latenza** che caratterizza ogni Cloud Provider

Conclusioni e sviluppi futuri

Conclusioni

- Progetto per **monitoraggio dinamico**, **analisi** e **presentazione** dei **ritardi di rete** dei principali **data center**, per i diversi fornitori di servizi Cloud
- Possibilità di **deployment world wide**

Sviluppi futuri

- Scaling orizzontale attraverso **sharding** del database per migliori prestazioni
- Deployment su **Microsoft Azure** e **IBM Cloud**
- Implementazione di ulteriori grafici per ampliare il potere espressivo

Grazie

Competitors

Esistenza di concorrenti che permettono di stimare la latenza.

Scenari e problemi principali:

- Possibilità di valutazione della latenza tra regioni, ma per un solo provider → **Nessuna unificazione e comparazione** tra fornitori
- Possibilità di analisi dei ritardi dal proprio terminale ad una regione, per tutti i providers → **Nessun riferimento** al modello ad **inter-data center**

Implementazione

Back-end

```
router.route('/pings/query/monthAggregation').get(async (req, res, next) => { // Definizione route
        //...
        Ping.aggregate()
        .allowDiskUse(true)
        .match({provider: myProvider})
        .project({month: {$month: "$timestamp"}, sameRegion: {$cmp: ['$from_zone', '$to_zone']}, timestamp: "$timestamp", time: "$time"})
        .match({$and: [{month: myMonth}, {sameRegion: mySameRegion }]})
        .group({_id : {"$dayOfYear": "$timestamp"}, avg: {$avg: " $time"}})
        .sort({_id: 1})
        .exec(function (err, resp) {
                if (err) {
                        console.log(err);
                } else {
                        res.json(resp);
                }
        })
});
```

- Definizione di **endpoint**
- Paginazione delle API
- Interrogazione database: **query**

Implementazione

Front-end

```
render(){
        return(
                <div>
                        {this.state.buttonClicked ? <MyGraph data = {this.state.graphData} graphType = {this.state.graphType} /> : null}
                </div>
        );
}
```

- Struttura a **componenti**
- Risultati visualizzati mediante **grafici**
- Comunicazione asincrona con il back-end