Framework and patterns facilitating cloud computing adoption

Ciprian Crăciun
Scientific adviser: Prof. Dr. Viorel Negru

West University of Timişoara, Romania

1 October 2010

Agenda

Introduction

- why **cloud computing**
- who are the involved actors (industry, academia, start-ups)
- what types of topics are there (philosophical, theoretical, engineering)
- what I have studied
- what I want to achieve

Agenda

Definition

NIST definition warning

"*Cloud computing is still an evolving paradigm. Its definitions, use cases, underlying technologies, issues, risks, and benefits will be refined in a spirited debate by the public and private sectors. These definitions, attributes, and characteristics will evolve and change over time.*" (NIST)

Definition

NIST definition

"*Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [...] on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service [...]*" (NIST)

Definition

A break in the clouds: towards a cloud definition

"*Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and / or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs.*" (Luis Vaquero)

Definition

... of what cloud isn't?

"*If the system requires human management to allocate processes to resources, it is not a cloud: it is simply a data centre.*" (Gianni Fenu)

Definition

... the most accurate one?

We can't definitively define cloud computing, but we can try analyzing and understanding it:

- scenarios
- technologies and challenges
- cloud mindset

Actors

- cloud provider
- cloud client / user
- final customer / user

Promises

Business model / operational benefits

- cost efficiency enabled by on-demand scaling
- simplified life-cycle management
- increased availability

Models

The way we solve problems

- `IaS` (*Infrastructure as a Service*)
- `PaS` (*Platform as a Service*)
- `SaS` (*Software as a Service*)

Applications

Q: What (kind of) problems are we actually supposed to be able to solve by using cloud computing?
A: ??? ... The same ones as before?!

Application taxonomy

The responsiveness axis

- on-line 24/7 applications
- batch processing

Application taxonomy

The predictability axis

- unpredictable loads — Slashdot effect, holiday peaks, etc.
- predictable / constant loads

Application taxonomy

The resource types axis

- computing
- raw storage — file systems, block devices, etc.
- structured storagedatabases
- synchronous communication
- asynchronous communication

Agenda

Focus

- the challenge
- the mindshift

Covered technologies

- distributed file systems
- distributed databases
- map-reduce
- asynchronous queues

Overlooked topics

- virtualization
- SLA (*Service Level Agreement*)
- cloud economy
- physical / virtual provisioning
- security

Distributed file-systems

Taxonomy

- unified namespace networked file-systems — **AFS**, **CODA**, **WWW**(??)
- parallel / clustered file-systems — `GFS` (*Global FS*), **OCFS**, Lustre
- (real, large scale) distributed file-systems — `GFS` (*Google FS*), **HDFS**, Ceph

Distributed file-systems

Working principle

Distributed databases

Taxonomy (1)

- replicated databases
- partitioned / sharded databases
- (real, large scale) distributed databases

Distributed databases

Taxonomy (2)

- columnar databases — Google BigTable, HBase, Hypertable, Cassandra
- key-value databases — Amazon Dynamo / S3, Riak, Voldemort, Membase
- document databases — CouchDB, MongoDB

Distributed databases

Consistency levels

- strong consistency
- weak consistency
- eventual consistency

Distributed databases

Eventual consistency

"*The storage system guarantees that if no new updates are made to the object eventually (after the inconsistency window closes) all accesses will return the last updated value. The most popular system that implements eventual consistency is DNS, the domain name system. Updates to a name are distributed according to a configured pattern and in combination with time controlled caches, eventually of client will see the update.*" (Werner Vogels)

Distributed databases

Eventual consistency solutions

- synchronization
- ignorance
- append only
- versioning
- parametrization

Distributed databases

CAP theorem

"*You can have at most two of these properties [consistency, availability, (tolerance of network) partitioning] for any shared-data system.*" (Eric Brewer)

Distributed databases

Constraints

- denormalization
- limited / no range queries
- (native) lack of arbitrary queries
- (native) lack of aggregates

Map-reduce

Working principle

Map-reduce

Steps

- partitioning
- mapping
- sorting
- reduce

Map-reduce

Characteristics / pitfalls

- parallelization
- composability
- input data locality
- support data minimization / locality
- side-effects

Asynchronous queues

Case studies

- Cloud architectures (Jim Varia) —

  `http://jineshvaria.s3.amazonaws.com/public/cloudarchitectures-varia.pdf`

- Building scalable, reliable Amazon EC2 applications with Amazon SQS — `http://sqs-public-images.s3.amazonaws.com/Building_Scalable_EC2_applications_with_SQS2.pdf`

- Seven lessons on scalability from Reddit (Steve Huffman) —

  `http://www.slideshare.net/carsonified/steve-huffman-lessons-learned-while-at-redditcom`

Asynchronous queues

Case studies — architecture / properties

Architecture

- low-latency front-ends
- heavy lifting back-ends

Properties

- scalability
- fault tolerance

Asynchronous queues

Why not classical job schedulers / batch processing

- smaller granularity
- dynamic workflows
- lower overhead

Agenda

- decoupling
- built-to-fail
- asynchronous communication
- eventual consistency
- automatic life-cycle management
- autonomy
- data locality
- appropriate data model

Agenda

Research direction

- cloud **SDK**
- building upon asynchronous queues
- distributed databases (as a backup plan)

Cloud SDK

Characteristics

- autonomous scaling
- generic
- efficient
- provider independent
- development environment independent
- developer centred
- smart monitoring

Cloud SDK

Activities

- identify cloud application patterns
- refine and classify the patterns
- identify support components
- architect and develop the components
- prove it

Building upon asynchronous queues

- applying **SEDA** into the cloud
- creating a meta-virtual machine

Research environment

- project — `mOSAIC` (*Open-Source API and Platform for Multiple Clouds*)
- close collaboration — Prof. Dr. Dana Petcu, Marian Neagul, Silviu Panica
- open cooperation — the local (**IeAT**, **UVT**) research team