Placement constraints for a better QoS in clouds

Extending BtrPlace to support typing

Mathieu Bivert
Tutor : Fabien Hermenier

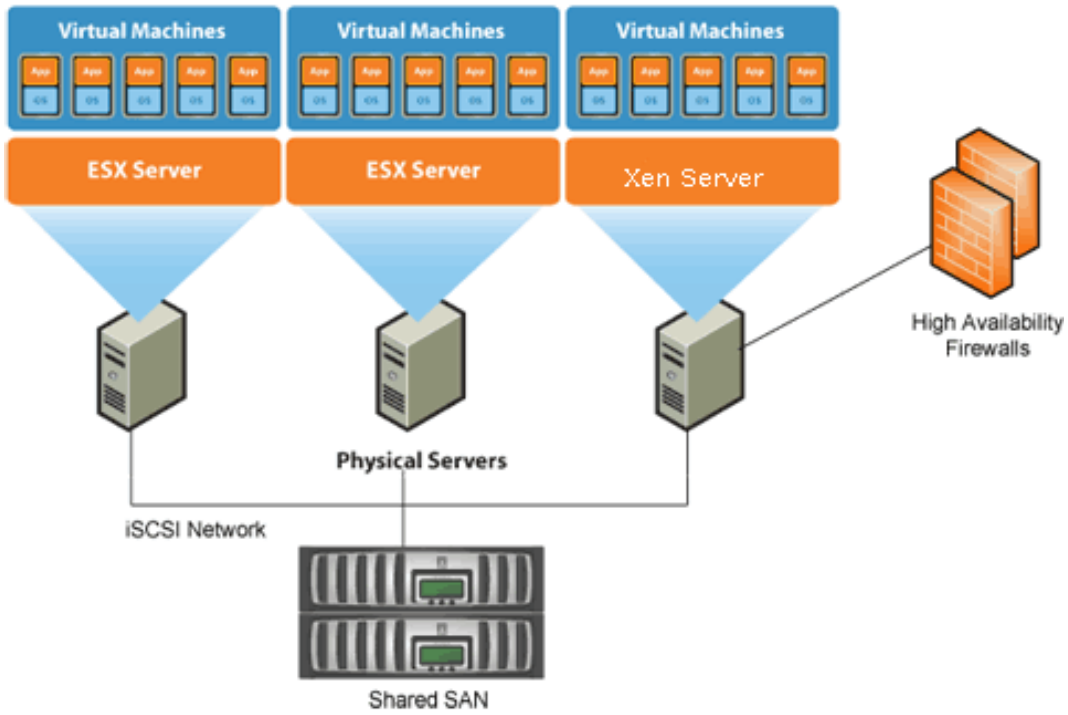
Polytech'Nice Sophia

March 8, 2013

Map

Introduction
Virtualized infrastructure
Virtualisation and Cloud
BtrPlace, a placement manager
Adding typing in BtrPlace
Modelisation in BtrPlace
Special case
General case
Additional constraints
Management
Problems encountered
Incomplete work
To sum up

Virtualized infrastructure



We define QoS as the performance, the avaibilitity, etc. provided by a cloud. Virtualization in clouds allows to

- Launch and stop services on the fly
- Replicates easily VMs running those services
- Facilitate administration

Clouds in business

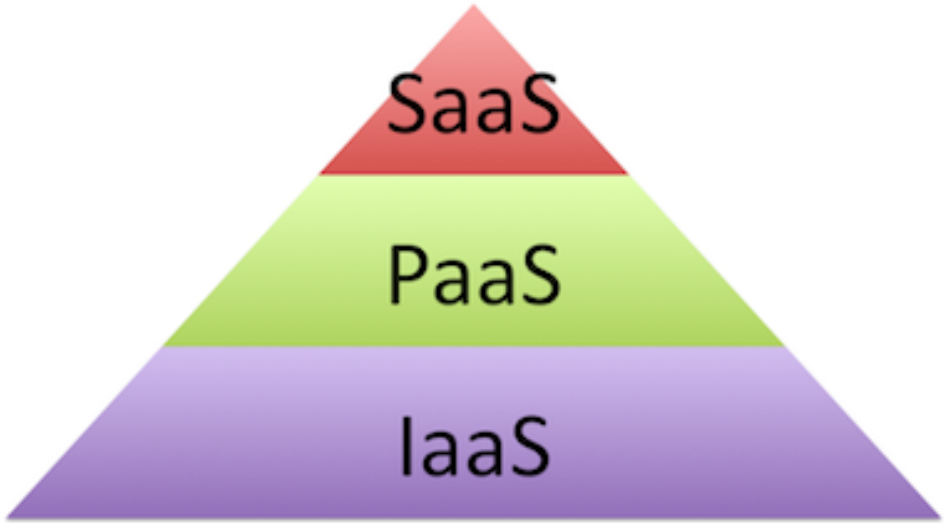
Large firms delegates their IT infrastructure to specialized companies

- Reduction of the costs (less hardware to buy and manage, less software to write, etc.)
- Augmentation of the QoS

However, by doing so, those firms:

- Lose control over their data
- Become dependent of another company

Different types of services



How is it done?

There are different kind of hypervisor,

How is it done?

There are different kind of hypervisor,
with different features,

How is it done?

There are different kind of hypervisor,
with different features,
and different licences;
Need for a software to manage at the Infrastructure level to place a set of virtual machines on a set of nodes.

How is it done?

There are different kind of hypervisor,
with different features,
and different licences;
Need for a software to manage at the Infrastructure level to place a set of virtual machines on a set of nodes. Would be usefull to teach this software about available hypervisor, their fonctionnality, etc.

BtrPlace

BtrPlace is a software that aimse to solve the problem of distributing a set of VMs on a set of nodes efficiently, by following some constraints, while being flexible. The constraints can be :

- imposed by the hardware, such as available ressources
- given by the user, following his needs (eg. replication of VMs)
- imposed by hypervisors licences

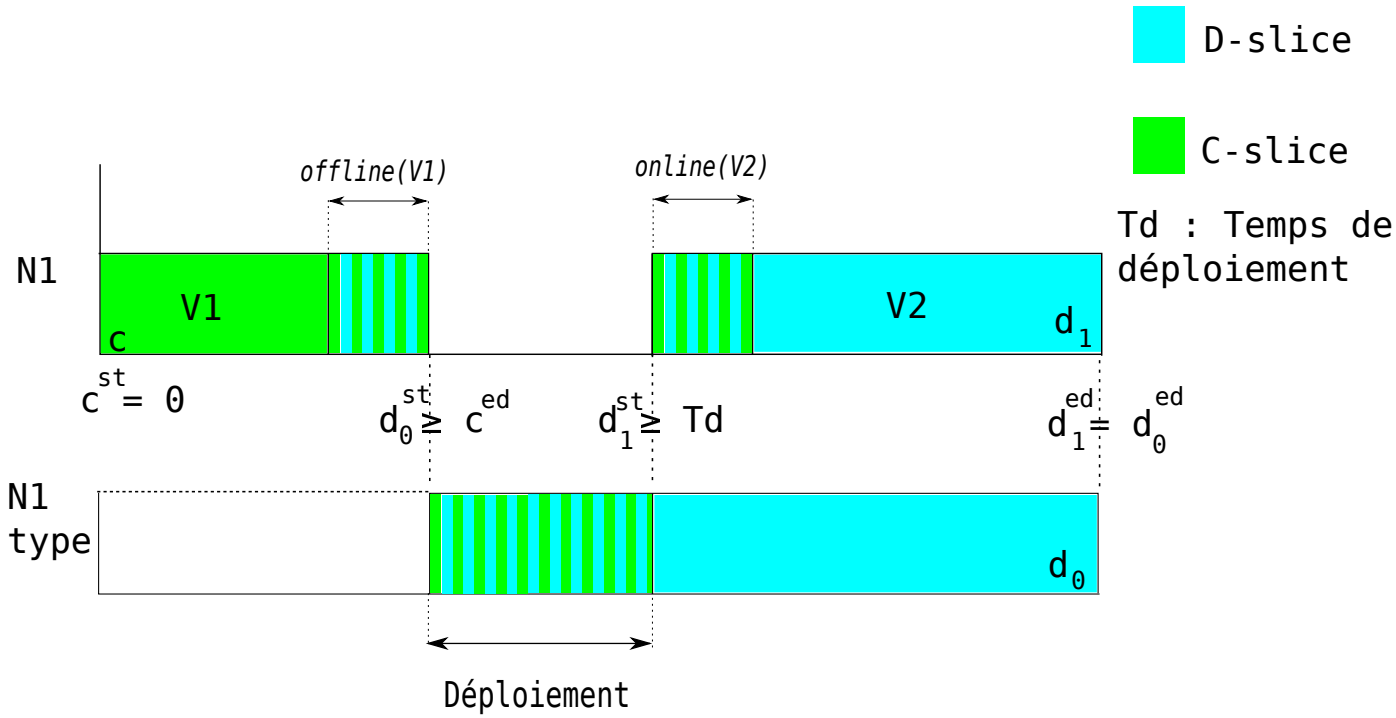
BtrPlace

BtrPlace is a software that aimse to solve the problem of distributing a set of VMs on a set of nodes efficiently, by following some constraints, while being flexible. The constraints can be :

- imposed by the hardware, such as available ressources
- given by the user, following his needs (eg. replication of VMs)
- imposed by hypervisors licences

As it competitors, BtrPlace doesn't make the distinction between different hypervisors. but designed to be extensible, so it should be reasonably easy to augment it to support typing.

Modelisation in BtrPlace



- **Type**, integer associated to each hypervisor
- **Deployment**, operation of rebooting a node and eventually changing its hypervisor
- **Reconfiguration**, operation during which BtrPlace change the placement of VMs on nodes following constraints

Proceeding of the work

We worked incrementally by

1. modeling and implementing a special case of the typing
2. modeling and implementing the general case
3. implementing some constraints associated to typing problems

Model

Hypothesis: we know which nodes are going to change their hypervisor, and the name of the new hypervisor.

Model

Hypothesis: we know which nodes are going to change their hypervisor, and the name of the new hypervisor.
For such a node, the following constraints must be satisfied:

$$P(c) = n \Rightarrow c^{\text{ed}} \leq D^{\text{st}}$$

$$P(d) = n \Rightarrow d^{\text{st}} \geq D^{\text{ed}}$$

Model

Hypothesis: we know which nodes are going to change their hypervisor, and the name of the new hypervisor.
For such a node, the following constraints must be satisfied:

$$P(c) = n \Rightarrow c^{\text{ed}} \leq D^{\text{st}}$$

$$P(d) = n \Rightarrow d^{\text{st}} \geq D^{\text{ed}}$$

Placement satisfied iff:

$$P(v) = n \Rightarrow T(n) = T(v)$$

Code

This special case is implemented through a constraint $Platform((n_i, h_j), (n_{i+1}, h_k), \ldots)$. There are two main methods in this class:

1. **inject**, which inject into Choco the two previous constraints
2. **isSatisfied**, which ensures the injected constraints are indeed satisfied in the new configuration

Model

BtrPlace should now deduce the new type of the nodes.
We add a vector v_i to each node. $v_i[t]$ represents the number of VMs running under the hypervisor t.

Model

BtrPlace should now deduce the new type of the nodes.
We add a vector v_i to each node. $v_i[t]$ represents the number of VMs running under the hypervisor t. The placement is satisfied iff:

$$(\exists! x \in v_i), x \neq 0$$

Currently, only the model has been defined correctly, no working code.

MinPlatform

MinPlatform(*nodes*, *type*, *n*) ensures at least *n* nodes from *nodes* runs hypervisor *type*.

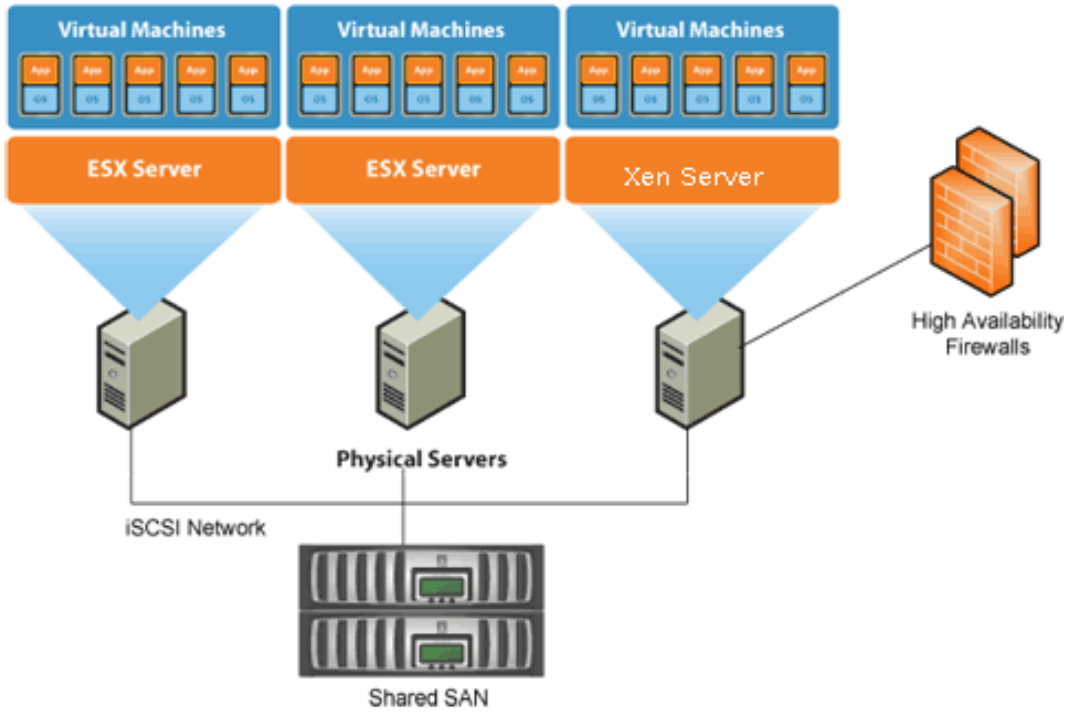


Figure : *MinPlatform*((*N*1, *N*2, *N*3), "*ESX*", 2)

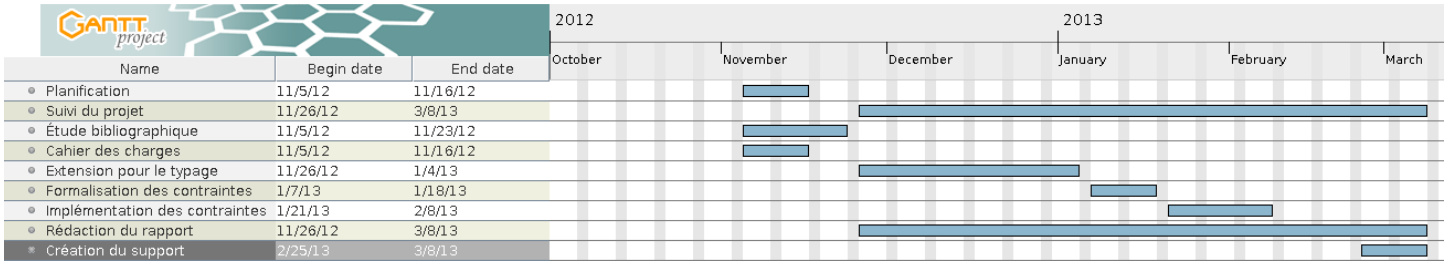
MaxVM

MaxVM(*nodes*, *type*, *n*) ensures at most *n* nodes runs hypervisor *type*.

MaxVM

MaxVM(*nodes*, *type*, *n*) ensures at most *n* nodes runs hypervisor *type*.
Other PFE project proposed by Fabien in response to licence limitations (VMWare notably) is easily implemented because typing is done with integer.

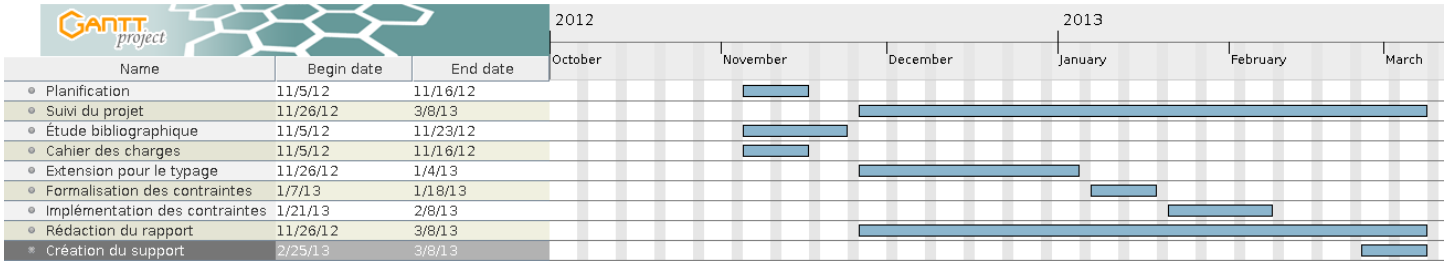
Timing management



Problem:

- not enough time spent on timing the work;
- incoherence in the DOW observed too late,

Timing management



Problem:

- not enough time spent on timing the work;
- incoherence in the DOW observed too late,

Possible Solution:

- spend more time on timing and structuring the work
- try to evaluate better exogenous elements (mainly other scholar works).

Complexity of BtrPlace

Problem : only documentation available : API Java.
Inadequate and insufficient to understand fully BtrPlace.

Complexity of BtrPlace

Problem : only documentation available : API Java.
Inadequate and insufficient to understand fully BtrPlace.
Possible solution : add two layers of documentation.

1. one describing the general structure of the software, with some example
2. and an other describing model generation and how to write simple constraints

What's done and what's missing?

Goals	State
Modelisation of the special case	done
Implementation of the special case	partial
Modelisation of the general case	done
Implementation of the general case	partial
Modelisation of new constraints	not done
Implementation of new constraints	mostly done

New competences and technologies

During this project, I learnt and revisited:

- Java and related tools (maven, IntelliJ, unit testing)
- Management of ressources and combinatorial problems
- Choco framework
- Git

How to improve what has been done and time estimation

- add more test for the special case (2 or 3 hours)
- implement general case at the right place, not as a user constraint, and tested (difficult to estimate)
- describe underlying models for constraints *MinPlatform* and *MaxVM*; complete and revised the implementation following the new models (1 or 2 hours)

Possible evolutions thanks to typing

Hypervisors licences and features can be pretty differents:

- some allows migrating VMs, some don't
- some put restrictions on usable hardware (number of NIC, RAM, CPU usable by the hypervisor)
- etc.

Typing could help modelize those limitations.

Questions

Thanks for your attention and time. Do you have any questions?