The importance of considering locality in P2P algorithms to operate distributed IaaS infrastructures

Jonathan Pastor, Marin Bertier, Flavien Quesnel, Adrien Lebre, Cedric Tedeschi

Departement of Computer Science, Ecole des Mines de Nantes
http://http://www.mines-nantes.fr/

Abstract. With the adoption of distributed cloud computing infrastructures, as the new platform to deliver utility computing paradigm, new algorithms for dynamical virtual machine scheduling leveraging on peer to peer approaches have been proposed.
Although it seems that these proposals tackle scalability issues by enabling the management of tens of thousands of VM upon hundreds of thousands of physical machines (PMs), none of them consider the effective costs of working on multi-site configuration.
To reduce these costs, one approach is to consider locality properties: servers collaborate first with close servers from the same geographical site. As a result, collaboration between servers can take into account infrastructure parameters such as bandwidth and response time and can be organized in a totally decentralized manner.
This paper illustrates the case of DVMS, a large scale virtual machine scheduler, that have been combined with a vivaldi based network overlay to leverage locality properties. Experimentations on the grid5000 tesbed provided promising results: this combination reach 83% of intrasite migration without source code modification, which provide a glimpse of the promising future of leveraging on locality properties to increase performance of massive distributed cloud platforms.

Keywords: Cloud computing, locality, peer to peer, network overlay, vivaldi, chord, DVMS, virtual machine scheduling

1 Introduction

2 Background

2.1 DVMS

- fundamentals
- limitations

2.2 P2P - locality

- Vivaldi
- active/lazy clustering

3 Contributions

3.1 Locality based overlay

- clustering
- Vivaldi + spirale

3.2 Dvms + PeerActor + locality

4 Experimentations

4.1 Implementation

A prototype of DVMS leveraging locality based overlay has been developed. The current version of DVMS are been developed over the PeerActor abstraction. PeerActor provides network abstraction that enables the design of distributed algorithm that are network overlay agnostic. We have developed two different overlay for the Peer Actor abstraction: Chord and a locality based overlay over Vivaldi.

The strength of this software architecture is that to enable an algorithm (like DVMS) to comply with a given network overlay, it only have to follow the Peer Actor API. This way, we were able to run DVMS over Chord or Vivaldi without any modification in its source code.

4.2 Grid5000' experiments

Objectives The prototype has been tested with a various number of experiments conducted on the Grid5000' testbed. The main objective of the experiments was to estimate impact of locality on the performance of a distributed scheduling algorithm. A significant portion of the reconfiguration time is spent in live migration of virtual machines, which depends of network parameters such as latency and bandwidth. One way to improve performance of distributed scheduling algorithm is is to promote collaboration between close ressources, which can be reach by maximising this ratio:

$$\frac{number\ of\ intrasite\ migrations}{number\ of\ migrations}$$

Experimental protocol For each experiment, we booked 40 compute servers spread on 4 geographical sites and 1 service server. The compute servers were used to run virtual machines and DVMS while the service node is used to stress several parameters of virtual machines.

Each compute node will host a number of virtual machines proportional to the number of CPU cores it has. In our case:

$$number\ of\ virtual\ machines\ =\ 1.3\ \times\ number\ of\ cores$$

Results The impact of locality on DVMS is significant: using a Vivaldi based network overlay leads to an average number of 83% of intrasite migrations while using a Chord based DVMS leads a ratio of 50% of intrasite migrations, as depicted in the following table:

network overlay	average number of intrasite migrations	average number of migrations
Vivaldi	83	100
Chord	40	80

Complete this section with more experimentation results.

5 Related work

5.1 DVMS

5.2 P2P

6 Conclusion