An Evaluation Testbed for Adaptive, Topology-Aware Deployment of Elastic Applications

Matthias Keller
Universität Paderborn
Warburger Strasse 100
D-33098 Paderborn
mkeller@upb.de

Christoph Robbert
Universität Paderborn
Warburger Strasse 100
D-33098 Paderborn
crobbert@mail.upb.de

Manuel Peuster
Universität Paderborn
Warburger Strasse 100
D-33098 Paderborn
peuster@mail.upb.de

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology—*Testbed*; C.2.4 [**Distributed Systems**]: Network operating systems—*Virtualization, Management, Testbed*

Keywords

Geographically Distributed; Adaptive Deployment; Testbed

1. INTRODUCTION

Cloud application providers who deploy their application at different cloud sites usually aim for close-by processing of user requests, benefiting from improved quality of service, and traffic reduction [4]. In this context, we dynamically scale applications to reduce costs by automating their deployment and adapting their resource allocation dynamically. We research the following questions: Where to allocate how many resources and how to apply the allocation? Which information is needed and how to exchange it? How can applications cope with ever changing resource allocations?

To practically evaluate our solutions, we created a flexible testbed. We share our insides and implementation to researchers tackling the diverse subproblems, various optimization goals, potentials for cost savings, and QoS improvements. We provide software with install instructions to construct your own private testbed [5].

Our testbed is two-layered: The bottom layer allows to test VM deployment on emulated, geographically distributed sites. It can be independently reused, is self-sufficient, and thus constitutes a small testbed on its own, the GeoDist Testbed (Section 2). The top layer allows to test adaptations and VM placement algorithms interactively or through predefined scenarios. Both layers together form the Adaptation Testbed (Section 3). Its capabilities are demonstrated in three different scenarios (Section 4).

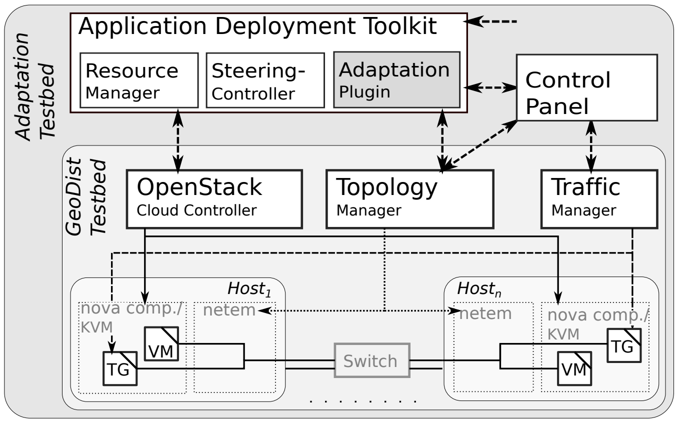

Figure 1: Two-layered Testbed architecture

2. GEODIST TESTBED

The testbed consists of hosts running the VMs managed by OpenStack [6] on a separate headnode. In order to emulate geographically distributed sites, we solved two issues: (i) To emulate different sites, the hosts are grouped; each host is assigned to one group representing a site. We configure OpenStack's availability zones, so we are able to specify the site a VM is deployed to (Figure 1). (ii) To emulate network properties between sites a Linux tool, called netem [2], is using. It builds upon Linux kernel's traffic control and alters sending behaviour of hosts so that latency, maximal data rate, jitter, or packet loss between hosts can be emulated. Netem rules can be applied for individual IP-addresses or subnets. The emulated properties between hosts are specified as a network topology, a graph with nodes representing sites/a group of hosts and annotated edges describing the network properties. Such a graph is automatically decomposed into netem rules for each host. However, this host-to-host configuration is insufficient as a VM's network is bridged and thus IP-addresses-based host configuration doesn't affect traffic with VM-IP-addresses. As a result, we need to maintain on each host VM-to-VM netem rules. Configuring these rules is not only a labour-intensive task but also changes very often – every time a VM is launched or destroyed. Thus, our Topology Module (TM) automatically generates the VM-to-VM rules and applies them on each host. Whenever the topology is re-set or the amount of VMs changes, the TM takes care about the reconfiguration. This additionally allows to emulate changing situations in the network (for instance, emulated congestion) while the VMs are running.

In line with a standard OpenStack installation, our testbed has two separate networks: one for infrastructure operations, image copying, or measurements and one for inter-VM, application-level traffic. The TM offers two level of control: The separate manipulation of both networks. We currently manipulate the application network as we measure application level effects. Other researchers interested in network properties and their relationship to infrastructure operations, like cross-site migration, can manipulate the infrastructure network as well.

The third module, the Traffic Manager, controls traffic generators deployed in small VMs at each host. In our current implementation, http requests are sent with a certain rate to specified VMs. The Traffic Manager can easily be extended to execute other requests.

3. ADAPTATION TESTBED

Building on top of the functionality provided by the bottom layer, two additional components realize a specialized testbed for adaptation algorithms: The Application Deployment Toolkit (ADT) and the Control Panel (CP).

The ADT provides a plugin mechanism for placement algorithms, offers a high-level interface to allocate geographically distributed resources as an allocation graph – similar to VXDL [3] – and automates common deployment tasks. The ADT collects topological data from the TM and provides them to the adaptation plugin, so topology-aware placement algorithms can use this data. The ADT, TM, and OpenStack realizes the geographically distributed deployment of application and their dynamic adaptation corresponding to changed load or utilization.

The other two components, the Traffic Manager, described in the GeoDist testbed (**??**), and the Control Panel, are tools to test the adaptive deployment and algorithms. The CP combines to aspects: It not only shows testbed states, e.g. which VM where, but also allows to change the framing conditions of test, e.g. how much traffic is generated where. CP's GUI is split into four parts: 1) A top view of how much VMs/resources are allocated where. 2) The adaptation plugin's internal data, upon which the allocation decision is based, is plotted. 3) The top view of how much load/traffic is generated from which sites. This view is interactive and allows live testing of the algorithm. 4) The measured QoS (in our case, response time) is plotted. Because of limited space, we omit a screenshot and redirect interested reader to our website [5].

4. STORYBOARD

During the demonstration, a two-tier application is deployed. It consists of two components: A frontend server answers requests of traffic generators. A backend server is partially asked. ADT automatically deploys and configures the servers in different VMs.

The allocated resources are adapted to improve average request response times while the situation changes: a) Load intensity changes at different sites; e.g., peak loads. b) Link property changes; e.g., emulated congestion. c) New Links are added; e.g., infrastructure investments. The individual changes are done live during the demonstration via the control panel; it also plots the effects on response times.

This will demonstrate three aspects: First, it shows a complete, working software stack, from applications in VMs to a cross-site resource management system. Second, it shows the behaviour of the monitor-reaction-loop in time. Third, an automated, dynamically adapted deployment and reconfiguration of multi-tier application architectures is realized.

The demonstration is performed with our **Cloud-in-a-Box:** We boxed 9 hosts with small form factor, commodity hardware in a compact, custom case. Each host has at least two network cards, which are interconnected by a 1 Gbit Switch.

5. RELATED WORK

Other testbeds for geographically distributed VM deployment are available: For instance, BonFire [1] or Amazon's Research Grants. While they offer resources from different sites, the network connecting these sites is the normal internet and thus not suitable for controlled measurements. A specialized site of BonFire offers network property configurations between VMs and aims to evaluate application behaviour with different connectivity scenarios. In contrast, we need to configure the network between sites as we aim for changing VM locations to improve application quality. Steiner et al. [7] showed a live monitoring of network properties between hosts/sites, which is used for adaptive deployment. Integrating their solution will enrich our testbed as currently the topology description is not only used to configure the network but also as direct input for placement algorithms.

6. ACKNOWLEDGEMENT

This research was supported by "Scalable and Adaptive Internet Solutions" (SAIL) project (funded by the european 7th framework programme; FP7-ICT-2009-5-257448; `http://www.sail-project.eu/`). Its developement is continued in "Comunicate. Green." (ComGreen) project (funded by Germany; BMWI's IT2Green Initiative; `http://www.communicate-green.de/`).

7. REFERENCES

[1] BonFIRE Project. `http://www.bonfire-project.eu/`. accessed 05.05.2013.

[2] Network Emulator (netem). `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`. accessed 24.04.2013.

[3] Koslovski et.al. VXDL: Virtual Resources and Interconnection Networks Description Language. *Networks for Grid Applications*, 2:138–154, 2009.

[4] Ibrahim et.al. Toward a new Telco role in future content distribution services. In *Proceedings of the 16th International Conference on Intelligence in Next Generation Networks (ICIN)*, pages 22–29. IEEE, 10 2012.

[5] University of Paderborn; Computer Network Research Group. Evaluation Testbed. `http://www.cs.uni-paderborn.de/en/research-group/research-group-computer-networks/people/matthias-keller-eng.html`. accessed 07.05.2013.

[6] OpenStack. OpenStack Cloud Software. `http://www.openstack.org/`. accessed 25.04.2013.

[7] Steiner et.al. Network-Aware Service Placement in a Distributed Cloud Environment. In *Proceedings of SIGCOMM Poster Session*, pages 73–74, 2012.