Sloth: SDN-enabled Activity-based Virtual Machine Deployment

Thomas Knauth ∗
TU Dresden

Pradeep Kiruvale
TU Dresden

Matti Hiltunen
AT&T Research

Christof Fetzer
TU Dresden

ABSTRACT
While cloud computing is excellent at supporting elastic services that scale up to tens or hundreds of servers, its support for small-scale applications that only sporadically require one VM is lacking. To better support this sporadic usage model, we employ Software Defined Networking (SDN) technology to expose events related to network activity. Specifically, we rely on notifications when switch flow entries are removed or missing to determine resource (in)activity. Our prototype, Sloth, activates virtual machines based on incoming network traffic. Conversely, idle VMs are suspended to conserve resources. We present the design and architecture of our SDN-enabled on-demand resource deployment solution. Our empirical evaluation shows that VMs can be activated in less than one second, triggered by SDN events. This on-demand resource activation opens up novel applications for Cloud providers, allowing them to transparently deactivate idle VMs while maintaining connectivity at the same time.

Categories and Subject Descriptors
D.4.7 [Organization and Design]: Distributed systems

Keywords
SDN; Cluster; Web Application; Data Center; Virtualization

1. INTRODUCTION

The cloud computing idea took the IT world in a storm and is now an established concept. The pay per use model and virtually infinite scaling potential made cloud computing attractive. However, cloud computing currently does not cater to customers with infrequent and sporadic resource needs. While some resources, e.g., network traffic, are metered and accounted for accurately, other resources, such as CPU and memory, are accounted for based on time instead of actual usage.

In previous work [2] we proposed to suspend idle virtual machines and only wake them up once the next request arrives. This effectively reduces the overall running time of VMs, cutting cost for the cloud customer, and freeing up resources at the cloud provider. The system’s design was based on an application-layer proxy to detect idleness. We sought a more general solution and found the mechanisms exposed by Software Defined Networking a suitable replacement. This poster paper describes our initial experience with a prototype named Sloth. Sloth uses notifications concerning flow entries as exposed by SDN to manage the virtual machine’s life cycle: suspending VMs when they are idle, and resuming VMs on incoming traffic.

2. ARCHITECTURE

The architecture for our Sloth prototype consists of an OpenFlow switch, an OpenFlow controller, and a set of servers functioning as virtual machine hosts. While it is possible to also control physical servers with Sloth, we are currently focusing only on managing virtual machines. The goal is to suspend VMs deemed idle and to resume them on incoming network traffic. Because we target virtualized resources in a data center, using network activity to determine system (in)activity is reasonable and has previously been explored [3]. Previous work, focusing on consolidation of idle desktop machines, used user input via the keyboard and mouse to detect idleness [1]. However, this is not applicable to virtual machines hosted in a data center, as the only means of interaction is over the network.

2.1 Virtual Machine Life Cycle

The OpenFlow controller manages the virtual machine life cycle in two ways: first, the controller suspends idle VMs as a result of expired flow entries. Second, a VM may be resumed when the controller detects a missing flow entry. Whenever a flow entry on an OpenFlow switch expires, the switch sends a flow removal message to the controller. A switch purges flow entries automatically because either the idle or hard timeout associated with the flow entry expired. As the name suggests, the idle timeout triggers a flow expiration after n seconds of inactivity, i.e., the switch has not forwarded any traffic related to this entry. The hard timeout, on the other hand, lets a flow expire n seconds after its installation, irrespective of activity.

Note, that a flow removal message does not automatically mean that the corresponding VM will be suspended. Be-
cause a VM can have multiple flows directed to it, in the case of microflows, i.e., separate flow entries for each client connection, only when removing the last flow can the VM be suspended. In the case of macroflows, i.e., handling all client connection with a single flow entry, there will at least be two flow entries, one for each direction (in and outgoing). The controller maintains per-VM flow counts and only suspends the VM once the flow count reaches zero.

Analogous to suspending VMs based on flow removal messages, the controller resumes VMs on packet-in messages. When the switch receives a packet for which no flow entry exists, it sends a packet-in message to the OpenFlow controller. A missing flow entry combined with a flow count of zero means that the corresponding destination VM is sleeping. That is, the controller must wake up the VM, before pushing a flow entry to the switch and forwarding the packet. Instead of forwarding the packet to a sleeping VM, the controller puts the packet into a FIFO queue and issues a command to resume the VM. Bringing the VM up takes anywhere from less than a second to a few seconds. Subsequent packets, arriving while the VM is resuming, are queued without issuing another wake up command to the VM host.

The controller realizes that the VM is up again by looking out for gratuitous ARP messages originating from the VM. This is a common feature of virtual machine monitors to help with re-establishing network connectivity after, e.g., a migration. Migrating a VM between different physical hosts, likely also changes the physical switch ports the VM is reachable on. Gratuitous ARP messages proactively inform the network infrastructure about the changed topology. The gratuitous ARP broadcast is visible to the controller as a packet-in message. Now that the VM is up, the controller forwards any queued packets to it.

3. EVALUATION

Our evaluation focuses on the end-to-end latency clients experience when connecting to a sleeping VM. We determined the resume speed for 8 different applications: 6 HTTP services, an SSH, and an email server. Because Sloth operates at the link-layer, it can handle arbitrary application-layer protocols, unlike our previous solution which relied on

an application-specific proxy to detect idleness [2]. Each data point represents the average of 10 runs. Figure 2 shows the end-to-end latency as measured at the client. We observe that the hot cache response times for all applications, except the Mediawiki, are between 0.6 and 0.7 seconds. Because essentially no data is read from disk, this presents the baseline response time when resuming a KVM-based virtual machine. Further reducing this baseline would require digging into the internals of QEMU/KVM, to identify potential optimization points. We leave this for future work. Complex application stacks, such as a full Mediawiki installation, impose an additional penalty even when resuming from a hot cache. In our case, the average response latency with a hot cache for the Mediawiki is twice that of the other applications at 1.4 seconds.

4. CONCLUSION

We demonstrated the feasibility to use the event notification mechanisms exposed by SDN to detect idleness at the network layer. Expiring and missing flow entries are the triggers we used to suspend or resume virtual machines.

Acknowledgments

This research was funded as part of the ParaDIME project supported by the European Commission under the Seventh Framework Program (FP7) with grant agreement number 318693.

References
