Flauncher and DVMS – Deploying and Scheduling Thousands of Virtual Machines on Hundreds of Nodes Distributed Geographically

Daniel Balouek, Adrien Lebre, Flavien Quesnel

Flauncher and DVMS

Deploying and Scheduling Thousands of Virtual Machines on Hundreds of Nodes Distributed Geographically

Daniel Balouek[1], Adrien Lèbre[2] and Flavien Quesnel[2]

[1] AVALON Research Group, École Normale Supérieure de Lyon / INRIA / LIP, France
[2] ASCOLA Research Group, École des Mines de Nantes / INRIA / LINA, France
firstname.lastname@inria.fr

Abstract. Although live migration of virtual machines has been an active area of research over the past decade, it has been mainly evaluated by means of simulations and small scale deployments. Proving the relevance of live migration at larger scales is a technical challenge that requires to be able to deploy and schedule virtual machines. In the last year, we succeeded to tackle such a challenge by conducting experiments with Flauncher and DVMS, two frameworks that can respectively deploy and schedule thousands of virtual machines over hundreds of nodes distributed geographically across the Grid'5000 testbed.

1 Introduction

Virtualization [6] enables people to use and manage distributed infrastructures in a more flexible way, by encapsulating the software in virtual machines (VMs).

One of the key features provided by this technology is the live migration of virtual machines [4], that consists in moving VMs from one node to another. Live migration has been used in particular (i) to evacuate the VMs hosted on nodes that are put into maintenance mode and (ii) to implement fine-grained scheduling policies dedicated to power management, workload consolidation or workload balancing.

Although live migration has been an active area of research over the past decade, it has been mainly evaluated by means of simulations and small scale deployments. Considering the increasing size of distributed infrastructures, it becomes critical to investigate the relevance of live migration on a larger scale, by performing experiments involving thousands of VMs and hundreds of nodes that might be geographically distributed across several sites. In the last year, we succeeded to perform these experiments on the Grid'5000 testbed by leveraging two frameworks, Flauncher and DVMS.

We describe Grid'5000, Flauncher and DVMS in the following sections.

2 Grid'5000

Grid'5000 is an experimental platform dedicated to experiment-driven research related to large scale distributed systems. It is composed of approximately 1200 nodes (8000 cores) distributed across 10 sites, mainly located in France. It provides support for experiments on all layers of the software stack.

More specifically, we used 4 key Grid'5000 services during our experiments: (i) the *OAR* batch scheduler, to manage resources on the platform, (ii) *Kadeploy*, to deploy our software stack on the nodes, (iii) *g5k-subnets* and (iv) *KaVLAN*, to configure the network respectively by reserving IP address ranges, and by isolating an experiment from the rest of the testbed using on-the-fly switches reconfiguration.

3 Flauncher

Flauncher [3] is a set of scripts leveraging the Grid'5000 software stack; it enables users to easily start a large amount of KVM VMs on several sites of the testbed.

These VMs can be used at user convenience, for example to investigate particular concerns, such as the impact of migrating a large amount of VMs simultaneously, or to study new proposals dealing with VM image management. To our knowledge, there is no such tool to quickly deploy and use a large amount of virtual machines.

3.1 Deployment cycle

The logical steps of a deployment are depicted in Figure 1 and described hereinafter.

Booking resources: The first script is in charge of finding available nodes that support hardware virtualization, booking them and requesting network resources, i.e. a /18 subnet for the IPs and a global VLAN (if needed); global VLANs are implemented by means of *KaVLAN* and enable us to virtualize several sites as a unique one, to avoid network domain issues when a VM is migrated from one network to another.

Deploying and configuring the physical nodes: This task consists in deploying the KVM hypervisor and the libvirt management stack on the worker nodes. During the deployment phase, the nodes are rebooted inside the global VLAN, if one has been booked during the previous step. At the end of the deployment, the global routing is configured on each node and the network is isolated from the usual routing policy. Moreover, the virtual machine disk images are copied on the worker nodes.

Creating and booting the virtual machines: The virtual instances are created and booted in parallel on all the worker nodes. Each virtual machine receives an IP address and a name by leveraging *g5k-subnets* and a round-robin assignment policy, which allows us to identify and communicate with all the virtual machines.

Worker nodes and virtual machines are checked at each step to detect potential failures; in that case, the step is performed again on the faulty components with a determined amount of retries. Communications are performed using tree-based tools; this approach ensures scalability and allows users to simultaneously run a script on multiple nodes and/or gather information.

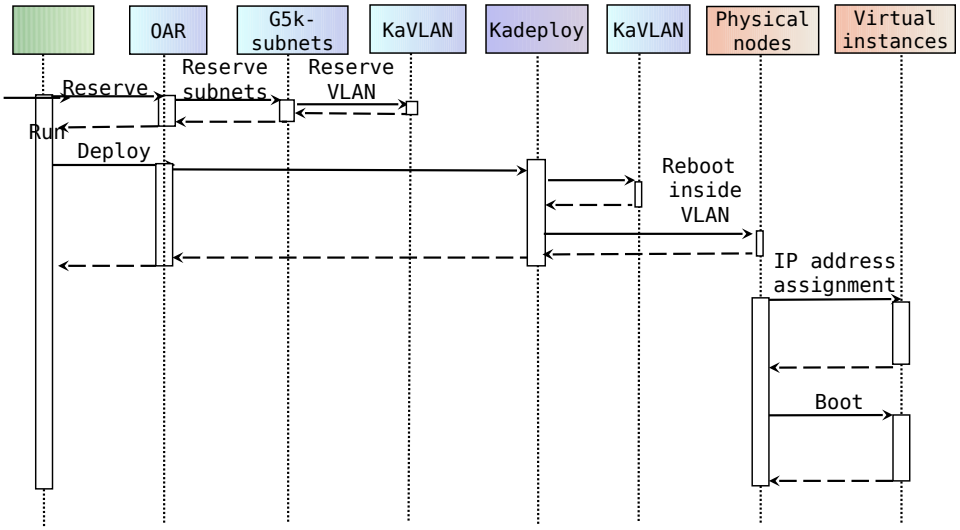


Fig. 1. Configuration steps performed by Flauncher to start the VMs on Grid'5000

3.2 Results

Figure 2 presents the duration of a full deployment depending on the number of physical machines (PM). Virtual machines are created and booted in parallel, thus the duration depends on the node that is the slowest to perform these operations.

Considering that (i) the worker nodes must support hardware virtualization to start KVM instances and that (ii) *KaVLAN* must be available on the sites, the largest experiment conducted up to now involved 10240 KVM VMs and 512 nodes distributed across 10 clusters on 4 different sites.

Flauncher has been used in particular to validate DVMS.

Experiment	251 PMs, 2000 VMs	309 PMs, 3325 VMs	467 PMs, 4754 VMs	512 PMs, 10240 VMs
Deployment + configuration (min)	17	20	32	35
VMs creation + boot (min)	15	15	15	15

Fig. 2. Experiments with Flauncher on the Grid'5000 testbed

4 DVMS

DVMS [7] (*Distributed Virtual Machine Scheduler*) is a framework that enables VMs to be scheduled cooperatively and dynamically in large scale distributed systems.

4.1 Overview

DVMS is deployed as a set of agents that are organized following a ring topology and that cooperate with one another to guarantee the quality of service (QoS) for the VMs.

When a node cannot guarantee the QoS for its hosted VMs or when it is under-utilized, it starts an iterative scheduling procedure (ISP) by querying its neighbor to find a better placement; it thus becomes the initiator of the ISP. If the request cannot be satisfied by the neighbor, it is forwarded to the following free one until the ISP succeeds. This approach allows each ISP to send requests only to a minimal number of nodes, thus decreasing the scheduling time without requiring a central point. In addition, this approach allows several ISPs to occur independently at the same moment throughout the infrastructure; in other words, scheduling is performed on partitions of the system that are created dynamically, which significantly improves the reactivity of the system. Communications are handled efficiently, as each node involved in a partition can forward a request directly to the first node outside its partition, by means of a "first out" relation.

An example involving three partitions is shown on Figure 3; in particular, we can see the growth of partition 1 between two steps.

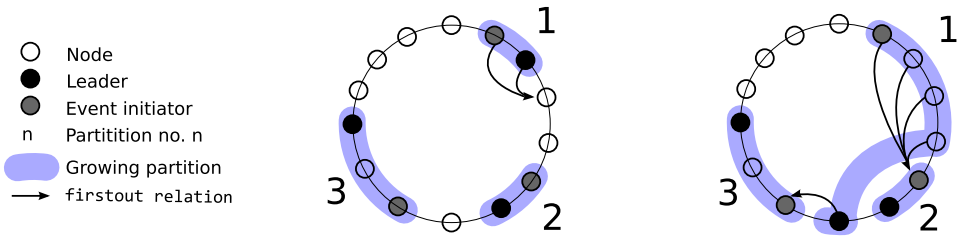


Fig. 3. Solving three problems simultaneously and independently with DVMS

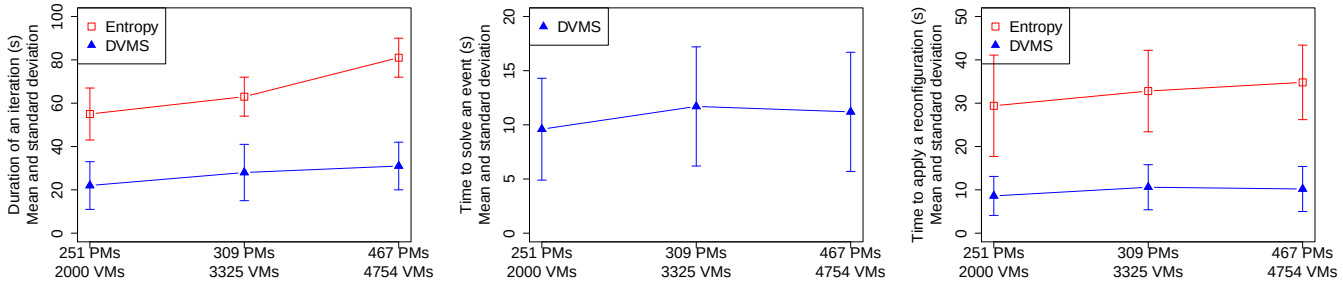
4.2 Experiments

We implemented a prototype in Java, and leveraged the scheduling policies used in Entropy [5], a centralized virtual machine scheduler.

We evaluated DVMS by means of experiments on the Grid'5000 testbed, involving up to 4754 VMs and 467 nodes (distributed across 4 sites). For comparison purposes, we performed the same experiments with Entropy. During these experiments, we used synthetic workloads (random CPU workload).

We observed that performing live migrations of virtual machines to solve overload problems in a large scale infrastructure was actually possible and meaningful. Moreover, we saw that a decentralized scheduler, such as DVMS, performed much better than a centralized one in this configuration. In particular, the results showed that:

- DVMS was able to solve a set of overload problems much faster than Entropy (cf. Figure 4(a)).
- DVMS solved each overload problem quickly (10 seconds on average, cf. Figure 4(b)).
- The time needed to solve a single problem (DVMS) and the time needed to solve all problems (Entropy) were dominated by the time needed to perform the migrations (cf. Figure 4(c)).



(a) Duration of an iteration (b) Time to solve an event with DVMS (c) Time to apply a reconfiguration

Fig. 4. Experiments with DVMS on the Grid'5000 testbed

To ensure that DVMS worked well at larger scales, we performed several simulations with the SimGrid toolkit involving 5K, 10K, 20K, 40K and 80K VMs (and respectively 0.5K, 1K, 2K, 4K and 8K nodes). These simulations confirmed that DVMS remained reactive whatever the scale was, while the centralized solution became inefficient at the largest scales.

It is worth noting that other works addressed the problem of decentralizing the dynamic scheduling of virtual machines (at least 1 hierarchical and 6 multi-agent approaches have been proposed so far) [7]. However, these approaches were evaluated (mainly by means of simulations) at smaller scales; therefore, we cannot conclude whether they were more reactive than DVMS or not.

5 Conclusion

In this document, we presented two frameworks, Flauncher and DVMS, that can respectively deploy and schedule dynamically thousands of virtual machines on hundreds of nodes of the Grid'5000 testbed.

Our experiments showed that both frameworks were scalable, and that DVMS was able to schedule VMs in a reactive way, whatever the scale was. These experiments are reproducible by leveraging the documentation of Flauncher [1] and DVMS [2]. As explained in this documentation, these tools are available on Grid'5000 on the Sophia site.

We are currently working on a new set of experiments involving the use of benchmarks, to inject a real load in the virtual machines and stress not only the CPU, but also the RAM, disk and network. We expect that these experiments will show that DVMS is able to manage resources on a finer-grained basis than traditional batch schedulers, much like distributed operating systems, without the complexity of development and maintenance.

References

1. Booting and Using Virtual Machines on Grid'5000. https://www.grid5000.fr/mediawiki/index.php/Booting_and_Using_Virtual_Machines_on_Grid%275000 (Mar 2013)
2. How to deploy Entropy and DVMS on Grid'5000. http://www.emn.fr/z-info/ascola/doku.php?id=internet:members:fquesnel:deploysimulatorg5kv3 (Mar 2013)
3. Balouek, D., Carpen Amarie, A., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., Lèbre, A., Margery, D., Niclausse, N., Nussbaum, L., Richard, O., Pérez, C., Quesnel, F., Rohr, C., Sarzyniec, L.: Adding Virtualization Capabilities to Grid'5000. Tech. Rep. RR-8026, INRIA (Jul 2012)
4. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: NSDI '05: Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation. pp. 273–286. USENIX Association, Berkeley, CA, USA (May 2005)
5. Hermenier, F., Demassey, S., Lorca, X.: Bin repacking scheduling in virtualized datacenters. In: CP '11: Proceedings of the 17th international conference on Principles and practice of constraint programming. pp. 27–41. Springer, Berlin/Heidelberg, Germany (2011)
6. Popek, G.J., Goldberg, R.P.: Formal requirements for virtualizable third generation architectures. Commun. ACM 17(7), 412–421 (Jul 1974)
7. Quesnel, F., Lèbre, A., Südholt, M.: Cooperative and Reactive Scheduling in Large-Scale Virtualized Platforms with DVMS. Concurrency and Computation: Practice and Experience (2012)