Emulation at Very Large Scale with Distem

Tomasz Buchert, Emmanuel Jeanvoine and Lucas Nussbaum
Inria, Villers-lès-Nancy, F-54600, France
Université de Lorraine, LORIA, F-54500, France
CNRS, LORIA - UMR 7503, F-54500, France
Email: `firstname.lastname@inria.fr`

***Abstract*—Prospective exascale systems and large-scale cloud infrastructures are composed of dozens of thousands of nodes. Evaluating applications that target such environments is extremely difficult. In this paper, we present an extension of the Distem emulator to allow experimenting on very large scale emulated platforms thanks to the use of a VXLAN overlay network. We demonstrate that Distem is capable of emulating 40,000 virtual nodes on 168 physical nodes, and use the resulting emulated environment to compare two efficient parallel command runners: TakTuk and ClusterShell.**

***Keywords*-large scale; emulation; VXLAN; virtualization**

I. Introduction

In the fields of cloud computing or high performance computing, the next generation platforms will involve more and more computing units. Being able to efficiently use those new infrastructures is a real challenge and numerous efforts are needed to write appropriate distributed applications. Furthermore, evaluating such applications can be complicated since the final platform is sometimes not available, at least for development purpose.

In previous work, we presented Distem [1], that provides an emulated platform on top of a regular cluster in order to facilitate the experimentation with distributed systems. In this paper, we focus on enhancing Distem to deal with large-scale experiments, typically those involving several dozens of thousands of nodes. In particular, we explain how leveraging overlay networks helps to push back some previously reached limitations. Being able to build large-scale platform is required to conduct large-scale experiments but is not enough. Indeed, we also present a methodology that leverages XPFlow [2] to perform clean and reproducible large-scale experiments.

The paper is organized as follows. Section II presents a general overview of Distem and how it has been enhanced to be able to emulate very large-scale platforms. Section III presents experiments carried out on 40,000 nodes that aimed at showing the scalability of two efficient parallel command tools. Section IV gives conclusions and presents future works.

II. Large Scale Distributed System Emulator

A. General overview

Distem [1] is a distributed system emulator that leverages advanced Linux features like LXC, CPU frequency scaling and traffic control, to emulate a heterogeneous platform on top of a homogeneous cluster. Heterogeneity can be obtained by (1) specifying a virtual network topology where latency and bandwidth of the links can be defined, (2) emulating a degraded CPU capability, (3) executing several virtual nodes on a single physical node. Furthermore, Distem is able to inject failures in the virtual platform in order to perform experiment in realistic conditions.

Distem is a valuable tool for distributed system evaluation since it allows one to perform experiments in various conditions with only one physical testbed. Furthermore, Distem improves reproducibility since the same experimental setup can be reproduced on different physical infrastructure. Typical applications are: the study of a peer-to-peer protocol, the study of a load-balancing algorithm, the study of a scheduler, etc.

B. Enabling Large-Scale Experiments in Distem

Previously, we have explored the Distem scalability to several thousands of virtual nodes (*vnodes*). Such virtual platforms can easily be set up on clusters with 100 nodes. Going beyond these numbers in terms of scale meets with various network issues. As a reminder[1], virtual nodes are connected together in the following way. Every *vnode* has a *Virtual Ethernet device* (veth) bridged into a Linux bridge created on the physical node (*pnode*). The network interface of the *pnode* is bridged into the Linux bridge. As a consequence, *vnodes* and *pnodes* are in the same L2 network. When running more than 5,000 *vnodes* using this architecture, several ARP-related problems appear. The ARP protocol is used to provide mapping between a network hardware address (i.e., MAC) and an IP address, a process required in Ethernet networks to enable communication between any two nodes. The encountered problems are:

1) the ARP protocol does not scale very well, in particular when a lot of requests are performed at the same time, a phenomenon known as *ARP flooding*;

[1]See [1] for a complete explanation.

2) the ARP tables, in particular those in the network equipment, have a hard-wired size limit (usually 8,096 or 16,192 entries) or are rather small (in the case of default Linux settings); consequently, the addresses of *vnodes* cannot be stored simultaneously in the table, leading to excessive numbers of ARP requests since the table is updated according to a LRU policy. In a related work on large-scale virtualization [3], the authors worked around this problem by partitioning the address into smaller networks, using routing between those smaller networks;
3) by default, the ARP entries are purged after a quite short time, leading to unnecessary and intrusive ARP traffic during an experiment.

Some of those issues has been fixed at the system level. Indeed, Distem performs tuning of various operating systems parameters on the *pnodes* to avoid useless ARP requests. In particular, on each *pnode*, Distem increases the ARP table size and the aging time of the ARP entries. Furthermore, Distem can be used to statically set a complete ARP table on all the *vnodes*, completely avoiding the ARP requests between them. These adjustments prove to be very effective, making it possible to perform experiments with 15,000 *vnodes* in the same L2 network. However, the problem of ARP tables still affects the network equipment, since in most of the cases it is not possible to modify their ARP settings.

To go beyond this limitation, we leveraged the capabilities of overlay networks, and in particular the VXLAN [4] protocol. VXLAN encapsulates L2 network traffic in UDP datagrams, which is very interesting in our context. Indeed, encapsulating the inter-*pnode* traffic using VXLAN overlay relieves the network switches since they do not have to deal anymore with *vnode* traffic directly: their ARP tables only need to store information about *pnodes*. We have explored a similar approach with GRE-tunnel encapsulation [5]. This is working well with a few *pnodes* in the network, but it does not scale at all since a full mesh between *pnodes* has to be created. Our experiments were unsuccessful with more than 10 *pnodes*.

Furthermore, to enable high bandwidth and low latency between *pnodes*, we used an InfiniBand network. This allows to lower the inter-*vnode* latency and to achieve large-scale experiments since the involved network equipment is more efficient when dealing with high-throughput traffic.

C. Technical details

Using VXLAN to encapsulate the inter-*pnode* traffic slightly modifies the previous Distem design. To create such an overlay, at least two possibilities exist: (1) leveraging OpenVSwitch[2] that offers VXLAN encapsulation, or (2) leveraging Linux that added VXLAN support since 3.7 version. OpenVSwitch is probably the best direction to create

[2]http://openvswitch.org/

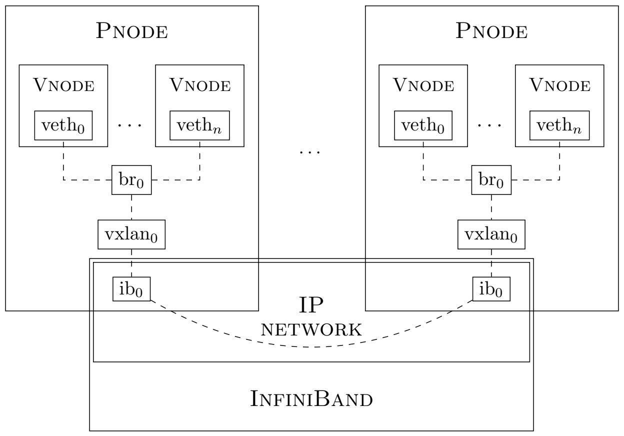

Figure 1. Network stack of a virtual node when using VXLAN encapsulation. Each virtual node has a virtual interface (veth$_i$) that is bridged with VXLAN interface (vxlan$_0$) on each *pnode*. This interface is associated with physical IP interface (ib$_0$). In our study IPoIB is used, but more traditional IP over Ethernet can be used as well.

virtual network, for instance it is used as a foundation of the network management of OpenStack. However, for simplicity of use and to minimize dependencies, we based Distem's VXLAN support on top of the Linux implementation.

Figure 1 presents the network stack of a *vnode* when using the VXLAN encapsulation in Distem. First, a VXLAN interface is set on top of the *pnode*'s physical interface. Because the VXLAN overlay carries aggregated traffic of all *vnodes*, a high-performance network interface offering IP interface, like 10G-Ethernet interface or IPoIB, is needed. Then, each *pnode* has a bridge that contains the VXLAN interface and interfaces of all *vnodes* on the given *pnode*. In this case, the bridge does not contain directly the physical network interface and all the *vnodes*' traffic is encapsulated before reaching the physical network interface. There is no need to specify routes between *pnodes* since VXLAN uses multicast-based discovery features that allow to establish the routes between several nodes automatically.

III. Experimental Validation

A. Purpose

In those experiments, we want to study two parallel commands tools: TakTuk [6] and ClusterShell [7], [8] (aka Clush). Both tools aim at efficient execution of command on a large set of nodes, which is a critical concern since executing administrative tasks or executing complex applications on large-scale clusters may rely on such tools. For instance, executing a simple command on dozens of thousands of nodes can take a lot of time if not performed in a proper way.

ClusterShell and TakTuk differ in their way to achieve high performance. ClusterShell uses a sliding window: the root node establishes SSH connections in parallel to several nodes, bounding the number of concurrent connections.

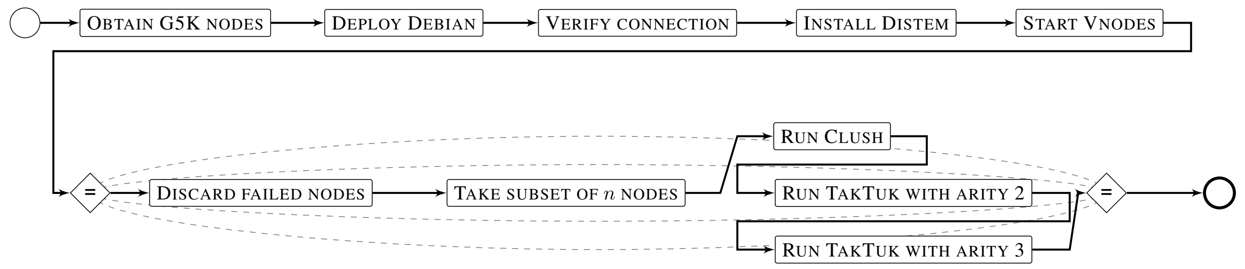

Figure 2. Experimental workflow. After the infrastructure is prepared (the first line of the workflow), a varying number of nodes (n) is used to run Clush and TakTuk methods in a simple loop (denoted with "="). At each iteration, the Distem infrastructure is analyzed and failed nodes are discarded if necessary.

TakTuk, instead, uses a tree-based algorithm, using nodes already connected to connect to additional nodes[3]. The experiment presented below will allow the comparison of those two strategies.

B. Physical Setup

The experiment has been executed on the Grid'5000 testbed [9]. More precisely, we used the *Graphene* and *Griffon* clusters from the Nancy site. *Graphene* is composed of 144 nodes (1 CPU Intel X3440 @2.53 GHz, 4 cores/CPU, 16GB RAM on each node) and *Griffon* is composed of 92 nodes (2 CPU Intel L5420 @2.50GHz, 4 cores/CPU, 16GB RAM on each node). Both clusters are interconnected with 20Gbit InfiniBand network and run Debian Jessie with Linux kernel at version 3.12.

On top of those clusters, we emulated a virtual platform with Distem and 162 physical nodes. Each *pnode* hosted 246 *vnodes*. All inter-*pnode* traffic was encapsulated inside a VXLAN overlay. Each VXLAN interface was plugged on top of an IPoIB interface to leverage the performance of the InfiniBand network.

C. Methodological Setup

Experiments where carried out with XPFlow [2], a tool based on business processes and workflows, and specifically designed to manage large-scale experiments. Among its features are robust failure handling and useful workflow patterns that model common experimental activities. In particular, XPFlow allowed us to transparently manage a failure of one physical node during our experiments, as it is explained in the following section. Figure 2 presents the experimental workflow.

Although our primary goal is to verify and show scalability of Distem, we wanted to show it using a study of various methods for command execution in large computer installations. To this end, we measured the time necessary to successfully execute the command `true` on a varying number of nodes. Each measure is repeated 3 times and the results are presented with 95% confidence intervals according to Student's t-distribution.

The raw results, the experimental workflow and associated files are available at http://www.loria.fr/~buchert/scale2014.tar.xz.

D. Results

The results with small number of nodes are presented in Figure 3. This range was chosen to show the size of infrastructure when sliding-window algorithms for command execution become inferior to ones based on tree topology. It happens around 1,400 nodes (for TakTuk with arity 3) and around 1,800 nodes (for TakTuk with arity 2).

Figure 4 shows the same type of experiments, but with a much larger infrastructure. Distem, thanks to it use of VXLAN encapsulation, shows a great scalability. Our goal of 40,000 nodes was almost reached, as we were able to successfully run our measurements with up to 39,852 virtual nodes. The reason for that is that one physical node failed during our measurements and had to be discarded. Fortunately, the presence of this failure was promptly detected and addressed by our experimental framework. Apart from that detail, we do not see why the scalability of Distem could not be pushed even more.

It can be expected that the total execution time consists of a constant factor, a linear factor (due to a constant time required by each node) and a logarithmic factor (due to a tree topology used by TakTuk). Therefore, we modeled the execution time using the model function

$$T\left(n\right) = A \cdot n + B \cdot \log\left(n\right) + C \qquad (1)$$

where T - execution time, n - number of nodes, and A, B, C - parameters of the model. The least squares fitting of the data to this model gives:

$$T_C\left(n\right) = 0.01649 \cdot n - 7.55 \cdot \log\left(n\right) + 52, \qquad (2)$$

$$T_2\left(n\right) = 0.00146 \cdot n + 3.55 \cdot \log\left(n\right) - 4, \qquad (3)$$

$$T_3\left(n\right) = 0.00099 \cdot n + 2.27 \cdot \log\left(n\right) + 4. \qquad (4)$$

where T_C, T_2, T_3 are results for Clush, TakTuk with arity 2, and TakTuk with arity 3, respectively.

[3]A similar mode of operation is available in ClusterShell's development branch, but we encountered problems when using it and discarded it from our experiments.

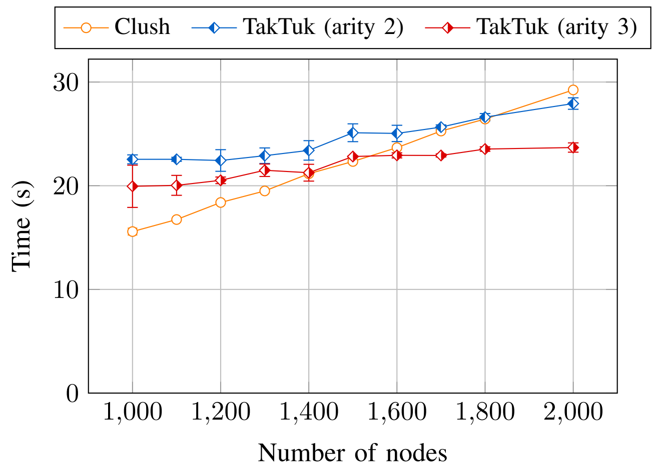

Figure 3. The time required to execute a command using various methods (small number of nodes). Clush performs better than TakTuk-based methods for around 1,400 nodes, but is outperformed by them for 2,000 nodes and more.

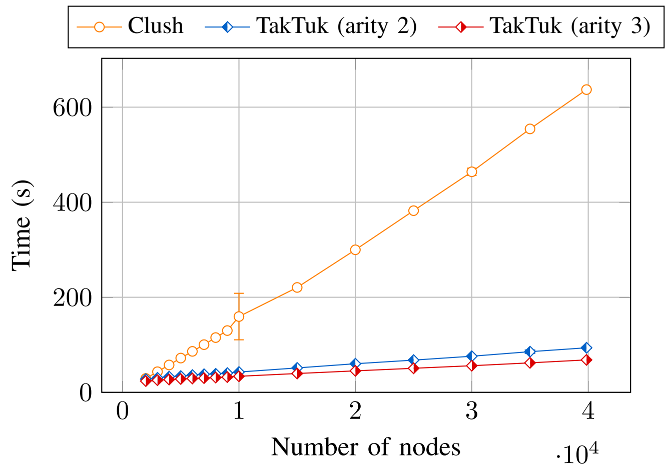

Figure 4. The time required to execute a command using various methods (large number of nodes). Clush is increasingly outperformed by both variations of TakTuk which use a tree overlay to execute commands.

We clearly see that the linear factor of Clush has the most impact on its total execution time, whereas in the case of TakTuk methods the linear factor is an order of magnitude smaller and constitutes a smaller percentage of execution time.

IV. Conclusions and Future Work

In this paper, we showed Distem's ability to scale to large emulated infrastructures. This feature depends on VXLAN encapsulation which is a modern network virtualization technology to address problems associated with large-scale distributed systems. We were able to successfully run our experiments with 39,852 virtual nodes hosted on 162 physical nodes interconnected with InfiniBand (although one node failed unexpectedly), showing scalability of VXLAN and Distem. Finally, we used this infrastructure to perform a simple analysis of different methods for command execution.

Future work will follow two directions. The first direction is to improve Distem in order to perform very large scale experiments with nodes in a single L2 network. Our next goal is to reach experiments involving more than 100,000 *vnodes*. Currently, one of the difficulties is to get access to large number of nodes in the same L2 network with root rights. The second direction is to enhance Distem with a distributed mode in order to perform experiments involving nodes from several geographical sites and many L2 networks. This way, Distem would be able to deal with extreme scale experiments.

Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] L. Sarzyniec, T. Buchert, E. Jeanvoine, and L. Nussbaum, "Design and Evaluation of a Virtual Experimental Environment for Distributed Systems," in *PDP2013 - 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, (Belfast, United Kingdom), pp. 172 – 179, IEEE, Feb. 2013.

[2] T. Buchert, L. Nussbaum, and J. Gustedt, "A workflow-inspired, modular and robust approach to experiments in distributed systems," Research Report RR-8404, INRIA, Nov. 2013.

[3] R. G. Minnich and D. W. Rudish, "Ten Million and One Penguins, or, Lessons Learned from booting millions of virtual machines on HPC systems," in *Workshop on System-level Virtualization for High Performance Computing in conjunction with EuroSys*, vol. 10, 2009.

[4] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks." Internet Draft, Feb. 2014.

[5] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)." RFC 2784, Mar. 2000.

[6] B. Claudel, G. Huard, and O. Richard, "TakTuk, adaptive deployment of remote executions," in *Proceedings of the 18th ACM international symposium on High performance distributed computing*, pp. 91–100, 2009.

[7] S. Thiell, A. Degrémont, H. Doreau, and A. Cedeyn, "ClusterShell, a scalable execution framework for parallel tasks," in *Linux Symposium*, p. 77, 2012.

[8] S. Thiell, A. Degrémont, and H. Doreau, "ClusterShell, Python library and tools for scalable cluster administration," in *PyHPC - Python in HPC Workshop*, 2013.

[9] "Grid'5000." http://www.grid5000.fr.