# Évaluation des performances des hyperviseurs pour l'avionique

Maxime Lastera[1,2], Eric Alata[1,2], Jean Arlat[1,2], Yves Deswarte[1,2], Bertrand Leconte[3], David Powell[1,2]
[1] CNRS ; LAAS ; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE, UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, France
[3] EYMI2 ; BP M3020 ; AIRBUS Operations SAS ; 316 route de Bayonne, F-31060 Toulouse Cedex ; France

***Résumé*—La technologie de virtualisation facilite la mise en œuvre d'architectures redondantes (voire diversifiées), afin de tolérer différentes classes de fautes et ainsi d'atteindre les objectifs de sûreté de fonctionnement visés. Cet article propose une méthodologie d'évaluation d'hyperviseurs afin d'analyser l'impact de la virtualisation sur la performance temporelle d'exécution. Plusieurs tests (CPU, mémoire, réseaux) ont été effectués, et des mesures du temps d'exécution ont été collectées sur différentes configurations mettant en jeu différents hyperviseurs.**

## I. Introduction

Le développement de systèmes critiques est soumis à de fortes exigences de validation et de vérification. Ces exigences ont un impact important sur les coûts de développement de ces systèmes. Par ailleurs, les composants pris sur étagère (*commercial off-the-shelf* ou COTS) offrent de nombreuses fonctionnalités tout en permettant de réduire ces coûts de développement. En revanche, ces composants n'ont pas été conçus dans le but de satisfaire les exigences liées au domaine avionique. C'est le cas des systèmes d'exploitation. Ces logiciels sont non seulement susceptibles d'être affectés par des fautes de conception accidentelles, mais sont aussi vulnérables à des attaques. Dans le domaine de l'avionique, les applications critiques sont généralement totalement séparées du monde ouvert, afin d'éviter toute interaction qui pourrait corrompre les systèmes embarqués à bord de l'avion. Cependant, les nouvelles générations d'avions ont besoin de plus d'interactions avec les systèmes au sol pour offrir des services étendus, et il est nécessaire de sécuriser les communications vis-à-vis des flux d'information potentiellement dangereux. Dans une précédente étude [1], l'utilisation de la virtualisation a été proposée pour assurer la fiabilité des applications critiques tout en permettant une communication bidirectionnelle entre les systèmes critiques embarqués dans l'avion et les systèmes moins critiques au sol. Afin d'analyser l'impact de l'utilisation de la virtualisation, nous avons ainsi développé un banc de test permettant de mesurer de façon précise des temps d'exécution d'architectures à base d'hyperviseurs. Plus précisément, nous nous intéressons à la capacité d'un hyperviseur à ne pas dégrader les ressources mises à la disposition des machines virtuelles. Dans cette étude, différentes configurations ont été expérimentées, allant du cas d'une machine dépourvue de système d'exploitation à celui d'une architecture complète avec un hyperviseur et un système d'exploitation exécuté dans une machine virtuelle.

La suite de cet article est organisée en cinq sections. La Section II décrit brièvement les travaux connexes à cette étude et met nos travaux en perspective dans ce contexte. La Section III présente brièvement le banc de test développé. La Section IV décrit le programme de test choisi pour l'évaluation de la mémoire. Ensuite, dans la Section V, nous détaillons les premiers résultats de l'évaluation. Enfin, la Section VI conclut cet article et présente les perspectives à envisager.

## II. Travaux connexes

L'évaluation des hyperviseurs a fait l'objet de plusieurs travaux. Dans [2], les auteurs ont comparé la dégradation de l'exécution d'une application entre une machine physique et différentes solutions de virtualisation. Ils utilisent le benchmark Ubench afin de réaliser leurs mesures. La machine physique est utilisée comme référence pour évaluer les différences entre les hyperviseurs considérés. Dans [3], différentes techniques de virtualisation sont comparées en prenant comme référence une machine physique munie d'un système d'exploitation non virtualisé. Dans ce cas c'est le benchmark VMmark qui est utilisé pour évaluer les performances. Le benchmark NAS est utilisé dans [4] afin de comparer les performances entre l'hyperviseur Xen et une solution VMware. Les auteurs étudient notamment l'influence du nombre d'unités de traitement (CPUs) virtuelles affectées à un hyperviseur. Les algorithmes RandomWriter et Sort sont utilisés dans [5] pour évaluer les performances d'un réseau de machines avec un hyperviseur. Dans ces différentes études, la mesure du temps d'exécution se fait de manière logicielle, notamment dans [5] la commande *top* est utilisée. Pourtant, dans un environnement virtualisé, l'horloge de l'hyperviseur est différente de l'horloge de la machine virtuelle. Aussi, un mécanisme de synchronisation doit être mis en place pour assurer l'ordonnancement de la machine virtuelle par l'hyperviseur. Or, d'une solution de virtualisation à une autre, ces mécanismes peuvent ne pas être identiques et de ce fait biaiser les mesures effectuées pour les temps d'exécution. Notre démarche se démarque de ces études par l'utilisation d'un périphérique matériel, indépendant de la configuration à évaluer. Cela nous permet à la fois de réaliser des mesures et d'observer le comportement des différentes configurations.

## III. Banc de test

Les mesures collectées doivent permettre de faciliter le choix d'un hyperviseur adapté aux exigences de performance. A cette fin des programmes de test distincts ont été définis pour surcharger différents composants matériels : mémoire, carte réseau et processeur. Un environnement de mesure spécifique a été développé pour permettre d'évaluer les temps d'exécution de ces programmes. Ces programmes sont exécutés

sur les différentes configurations de l'architecture, de manière incrémentale. Tout d'abord, nous utilisons une configuration minimaliste permettant uniquement un accès matériel, cette configuration est nommé « Pépin ». Ensuite, nous avons choisi deux versions du noyau Linux correspondant à celles utilisées par les hyperviseurs considérés. Il s'agit du noyau 2.6.32 utilisé par l'hyperviseur Xen et du noyau 2.6.34 utilisé par l'hyperviseur Qubes. Ces configurations testées sont nommées respectivement « 2.6.32 », « 2.6.34 », « Xen » et « Qubes » (*cf.* table 1). Nous avons utilisé une machine virtuelle exécutant un système d'exploitation pour représenter la dernière configuration. Une machine virtuelle est exécutée sur l'hyperviseur Xen et nommée « Xen-vm » et une autre machine virtuelle est exécutée par l'hyperviseur Qubes, nommée « Qubes-vm ». Ces deux machines virtuelles utilisent le même système d'exploitation, une distribution Fedora 13 avec un noyau 2.6.34.

## IV. Les programmes de test

Nous souhaitons évaluer la dégradation des hyperviseurs vis-à-vis des performances de plusieurs types d'applications qui peuvent être déployés dans l'environnement. Par conséquent, nous avons développé des programmes types qui sollicitent chacun un composant particulier du matériel (CPU, mémoire et réseau) [1]. Les applications sont développées de la manière suivante, écriture de *un* sur un bit du port parallèle avant le lancement de l'application puis écriture de *zéro* sur le même bit du port parallèle à la fin de l'exécution, ainsi le temps qui s'écoule sur le palier haut mesuré à l'oscilloscope représente le temps d'exécution de la charge. Par manque de place, dans la suite, nous ne traiterons que du test relatif aux transferts mémoire. Le programme de test mémoire est réalisé en langage assembleur pour nous permettre de maîtriser les instructions exécutées par l'application. Le test mémoire doit permettre d'identifier si l'hyperviseur exploite pleinement les capacités du bus FSB (*Front-Side Bus*). Ce dernier permet de connecter le processeur à la mémoire. Dans la section suivante nous présentons les résultats du test mémoire.

## V. Résultats

La taille mémoire à copier a été définie à 10Mo et répétée dix fois afin de mettre en défaut les différents caches. La copie mémoire représente donc un volume de 100Mo. La taille de l'échantillon d'étude a été fixée à mille expériences. Les histogrammes de la figure 1 correspondent aux valeurs moyennes calculées sur mille expériences pour chacune des configurations. L'échelle temporelle est exprimée en millisecondes. La table I présente les valeurs extrêmes correspondant à un intervalle de confiance à 95% pour les données que nous avons analysées. Nous observons que la configuration « Pépin » n'est pas optimisée, en effet elle a été conçue seulement pour accéder au port parallèle, aucune mise en place de cache n'est prévue dans cette configuration. Il existe entre les deux versions du noyau une différence qui peut s'expliquer par les améliorations apportées lors des changements de version du noyau Linux. Au niveau des hyperviseurs, Xen se comporte comme la configuration « Pépin », ce qui suggère qu'il n'implémente pas le mécanisme de mise en cache pour améliorer les transferts mémoire. Il a été conçu pour assurer la mise en place de techniques de virtualisation et d'isolation. En effet, la performance de la configuration « Xen-vm » est du même ordre que celle des systèmes d'exploitation non virtualisés (noyau 2.6.32 et noyau 2.6.34). L'hyperviseur Qubes suit une autre hypothèse. Il semble mettre en place des mécanismes de cache pour améliorer le transfert mémoire, car il obtient le meilleur résultat du test. Cependant la gestion de sa machine virtuelle est moins performante vis-à-vis de Xen. En effet, sur le test considéré, la configuration « Qubes-vm » est moins performante que la configuration « Xen-vm ».

1. Un programme complexe peut être vu comme la somme pondérée de ces trois applications types. Par conséquent, la dégradation de performance sera une somme pondérée de ces trois applications.

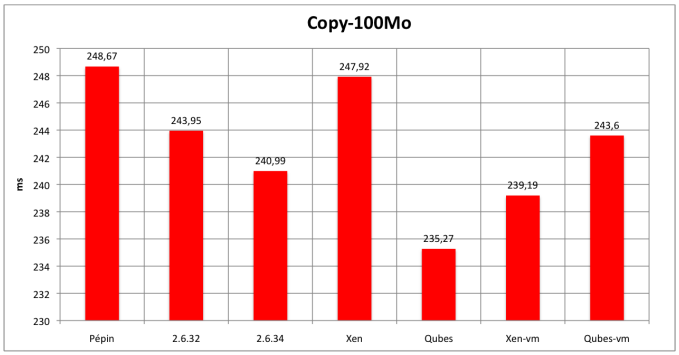


FIGURE 1. Histogramme comparatif

| Configuration | Pépin | 2.6.32 | 2.6.34 | Xen | Qubes | Xen-vm | Qubes-vm |
|---|---|---|---|---|---|---|---|
| Borne sup. | 248,71 | 244,59 | 241,01 | 247,94 | 235,3 | 239,21 | 243,63 |
| Borne inf. | 248,64 | 243,3 | 240,97 | 247,90 | 235,24 | 239,17 | 243,57 |

TABLE I
BORNE DE L'INTERVALLE DE CONFIANCE À 95%

## VI. Conclusion

Cet article présente une méthodologie d'évaluation des performances de différents hyperviseurs. L'utilisation d'un accès matériel tel que le port parallèle permet d'être indépendant des configurations à évaluer. Les résultats ont montré que les hyperviseurs ont chacun leur politique d'ordonnancement vis-à-vis de leur machine virtuelle. L'hyperviseur Xen semble privilégier sa machine virtuelle au dépend de ses propres performances alors que l'hyperviseur Qubes privilégie ses propres performances. Dans la suite de nos travaux nous essaierons d'appliquer cette méthodologie à d'autre hyperviseurs tels que OKL4 microvisor ou PolyXene.

## Références

[1] Y. Laarouchi, "Sécurités (immunité et innocuité) des architectures ouvertes à niveaux de criticité multiples : application en avionique," Thèse de Doctorat, LAAS-CNRS, INSA de Toulouse, Nov. 2009.

[2] X. Xu, F. Zhou, J. Wan, and Y. Jiang, "Quantifying Performance Properties of Virtual Machine," in *International Symposium on Information Science and Engineering (ISISE'08)*, vol. 1, 2008, pp. 24–28.

[3] R. McDougall and J. Anderson, "Virtualization Performance : Perspectives and Challenges Ahead," *ACM SIGOPS Operating Systems Review*, vol. 44, pp. 40–56, Dec. 2010.

[4] D. Bhukya, S. Ramachandram, and A. R. Sony, "Evaluating Performance of Sequential Programs in Virtual Machine Environments Using Design of Experiment," in *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2010, pp. 1–4.

[5] M. Kontagora and H. Gonzalez-Velez, "Benchmarking a MapReduce Environment on a Full Virtualisation Platform," in *Intl. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS)*, 2010, pp. 433–438.