Evaluation des performances réseau dans le contexte de la virtualisation XEN

Pascale Primet, Olivier Mornard, Jean-Patrick Gelas

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS
LYON-UCBL n° 5668

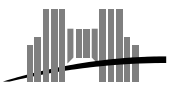
Evaluation des performances réseau dans le contexte de la virtualisation XEN

Pascale Primet Olivier Mornard Jean-Patrick Gelas

Décembre 2007

Rapport de recherche N° RR2008-01

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`



Evaluation des performances réseau dans le contexte de la virtualisation XEN

Pascale Primet Olivier Mornard Jean-Patrick Gelas

Décembre 2007

Abstract

Virtualization technics of the Operating System improve security, isolation, reliability and flexibility of the environments. These technics become a must in large scale distributed system and begin to be used in data transport networks. However, virtualization introduced an overhead which must be integrated to virtualized system performance models in order to forecast their behavior. In this article, we analyse this overhead and give an experimental evaluation of the virtualization impact with XEN over the network performance in high speed network context. We show that the throughput reached with TCP is barely affected by the virtulization at the expense of a growing CPU consumption. Moreover, we study the conflicts between protocol process and standard process to access CPU resources. Finally, we show the impact of the TCP's congestion window size in case of several virtual machines running.

Keywords: Network performances, virtualization, Xen

Résumé

Les techniques de virtualisation des systèmes d'exploitation améliorent la sécurité, l'isolation, la fiabilité et la flexibilité des environnements. Ces techniques deviennent incontournables dans les systèmes distribués large échelle et commencent à être exploitées dans les réseaux. Cependant la virtualisation introduit un surcoût devant être intégré aux modèles de performance des dispositifs virtualisés afin d'en prédire le comportement. Dans cet article nous analysons ce surcoût et évaluons expérimentalement l'impact de la virtualisation XEN sur les performances réseau dans le cas des réseaux haut débit. Nous montrons que le débit obtenu par TCP est très peu affecté par la virtualisation au prix d'une augmentation de la consommation CPU. Nous étudions les conflits entre le traitement protocolaire et les processus de calcul pour l'accès aux capacités des processeurs. Nous montrons aussi l'impact de l'ajustement de la taille de la fenêtre de congestion de TCP dans le cas de plusieurs machines virtuelles.

Mots-clés: Performances réseaux, virtualisation, Xen

1 Introduction

La virtualisation a été proposée très tôt dans l'histoire de l'informatique notamment par IBM Research Centers, dans le milieu des années 1960. L'idée générale est de permettre de lancer de multiples instances de système d'exploitation sur le même matériel. Ainsi chaque instance de système d'exploitation virtuel se comporte fonctionnellement exactement comme si elle était en cours d'exécution seule sur le même matériel. Aujourd'hui, la virtualisation de l'OS remporte un immense succès car cette technique permet d'améliorer la sécurité, l'isolation, la fiabilité et la flexibilité des environnements. Ainsi, ces techniques deviennent incontournables dans les systèmes distribués large échelle dans lesquels ces défis doivent être relevés. Dans le domaine des réseaux, le phénomène est plus récent et plusieurs projets explorent l'intérêt d'utiliser les techniques de virtualisation dans les routeurs eux-mêmes. Les fabriquants de processeurs fournissent aujourd'hui en standard un jeu d'instructions processeur (virtualisation matériel) permettant d'isoler efficacement les machines virtuelles, et ce même dans les machines grand public.

Les principaux objectifs de la virtualisation sont les suivantes :

- fournir un environnement confiné dans lequel des applications non fiables (au sens confiance du terme) peuvent être exécutées ;
- limiter l'accès et l'utilisation des ressources matérielles par le biais de techniques d'isolement ou les élargir de façon transparente pour les applications ;
- adapter l'environnement d'exécution à l'application au lieu de porter l'application sur chaque environnement d'exécution ;
- utiliser des mécanismes OS optimisés et dédiés (ordonnanceur, gestion de la mémoire virtuelle, protocole réseau) pour chaque application ;
- gérer globalement les applications et les processus en cours d'exécution dans une machine virtuelle. Par exemple, la répartition du temps peut être différente pour chaque machine virtuelle et sous le contrôle d'une machine virtuelle gestionnaire.

Dans le cas des systèmes distribués large échelle, on utilise la virtualisation pour se protéger contre les programmes malveillants, pour éviter les conflits, pour assurer la sécurité de l'exécution en utilisant un environnement certifié, pour mesurer la consommation de ressources. La virtualisation permet aussi de protéger l'exécution de longs travaux en fournissant des points de récupération et la possibilité de redémarrer à partir d'un certain état sauvegardé éventuellement sur un autre système physique. La virtualisation est très utile quand le redimensionnement de l'environnement est nécessaire par exemple en cas de modification des besoins en matière de capacité. Ainsi au cours de l'exécution de la tâche, des capacités supplémentaires peuvent être requises (par exemple la capacité de CPU, la capacité de mémoire, bande passante d'E/S, etc.). Si le système physique est en mesure de répondre aux besoins accrus, ces nouvelles capacités peuvent être fournis localement, sur le même système physique. De même, en cas de modification de la capacité offerte et disponible, la virtualisation permet de redimensionner l'infrastructure sans affecter fonctionnellement les applications en cours. La capacité disponibilité dans le système physique (capacité de CPU, mémoire, bande passante, etc.) peut augmenter par exemple en raison de ressources récemment libérées. Enfin la virtualisation permet d'optimiser l'utilisation d'un ensemble de ressources par équilibrage de charge (*load-balancing*)

des machines virtuelles sur des machines physiques. Ainsi les machines virtuelles peuvent migrer au moment de l'exécution pour des raisons d'économie d'énergie, de maintenance et d'entretien, d'adaptation aux besoins en matière de capacités physique, d'opportunités d'accès à de nouvelles ressources.

L'utilisation de la virtualisation système dans le cas des routeurs, permet aussi d'augmenter la sécurité, de protéger des trafics, d'adapter la gestion des flux en fonction des ressources, de la consommation d'énergie, d'adapter le système et les processus de routage aux trafics et non l'inverse...

La virtualisation offre donc de nombreux avantages et de nombreuses opportunités pour résoudre aussi bien des problèmes de calcul que de communication, mais elle apporte nécessairement une contre-partie. Le principal inconvénient est son coût en termes de consommation de ressource qui peut être plus ou moins déterministe. Alors que le surcoût et le passage à l'échelle des techniques de virtualisation a été bien étudié au niveau système d'exploitation, peu de travaux ont été menés dans le domaine de l'analyse des performances réseau de bout en bout. Nous nous attachons donc dans ces travaux à évaluer et à comprendre le surcoût que représente la virtualisation en terme de consommation de bande passante réseau et de traitement protocolaire, en particulier dans les réseaux haut débit et les environnement haute performance. En effet, ce surcoût doit être intégré aux modèles de performance des dispositifs virtualisés (cluster virtuels ou routeurs virtuels) afin d'en prédire le comportement. La section 2 propose un bref état de l'art des différentes méthodes de virtualisation, puis se concentre sur les mécanismes d'accés au réseau fournis par les principales solutions de virtualisation actuellement disponibles. Nous développons dans la section 3 une analyse de l'implémentation des protocoles et des mécanismes réseau dans le cas de la solution XEN. Enfin la section 4 propose un protocole expérimental pour évaluer les performances réseau obtenues dans le contexte du protocole TCP et de l'environnement XEN. Nous étudions le débit obtenu ainsi que la consommation CPU au niveau des connexions TCP, les conflits entre le traitement protocolaire et les processus de calcul pour l'accès aux capacités des processeurs ainsi que l'impact de l'ajustement de la taille de la fenêtre de congestion de TCP. La dernière section 6 développe la conclusion et quelques perspectives.

2 Les mécanismes de virtualisation

Afin de comprendre les interactions entre le réseau et les techniques de virtualisation d'OS, cette section présente un aperçu des différentes approches existantes. Nous considérons les propositions qui ont été faites : au niveau application, les émulateurs, le système d'exploitation dans l'espace utilisateur, la paravirtualisation, la virtualisation assistée matériellement.

Virtualisation au niveau application : L'approche de la virtualisation au niveau application permet de virtualiser de multiples serveurs isolés et sécurisés sur un seul serveur physique. Il n'y a pas de système d'exploitation dans les machines virtuelles. Tous les serveurs virtuels partagent le même système d'exploitation : celui de la machine physique. Cependant, les applications ne percoivent qu'un seul système d'exploitation, comme si elles ne partageaient pas le serveur physique. Beaucoup de projets tel VServer sont basés sur cette approche.

Émulateurs : Dans l'approche émulateur, une machine virtuelle est simulée ainsi que tout son matériel spécifique. Il est donc possible d'utiliser un système d'exploitation classique non modifié sur cette architecture. On peut également émuler un processeur différent de celui de la machine hôte. Cette technique est couramment utilisée pour développer des logiciels pour les nouveaux processeurs avant qu'ils ne soient effectivement disponibles.

Système d'exploitation dans l'espace utilisateur : Une dernière possibilité de virtualisation est d'exécuter le système d'exploitation dans l'espace utilisateur comme une application classique. Cette solution n'est pas très performante et est principalement utilisée pour les développements des noyaux des systèmes. Le plus célèbre projet basé sur cette approche est l'UML ou User Mode Linux.

Paravirtualisation : Par rapport à l'approche précédente, la technique de paravirtualisation [13] ne nécessite pas de simuler le matériel, mais plutôt d'offrir une API offrant les mêmes services que le matériel. Ceci implique une modification du système invité. Par conséquent, les performances peuvent être améliorées par rapport à l'approche précédente. La série d'abstractions (processeur virtuel, mémoire virtuelle, réseaux virtuels, etc...) permet à différents OS d'être portés. Ces abstractions ne sont pas nécessairement similaires à de réelle ressources matérielles de la machine hôte. Ces services sont fournis par des appels système spéciaux effectués auprès d'un hyperviseur. Le projet le plus connu basé sur cette approche est Xen.

Xen [3, 9] est un moniteur de machines virtuelles pour l'architecture x86, qui permet des exécutions simultanées de plusieurs OS en fournissant une isolation des ressources et un confinement des différents environnements d'exécution. Dans Xen, un domaine 0 (Dom0), créé sur l'ordinateur hôte au démarrage, accède à l'interface de contrôle et exécute l'application de gestion des domaines virtuels. Le domaine 0 permet de contrôler le lancement et l'arrêt des autres domaines dans lesquels les OS invités s'exécutent. Pour mettre en œuvre la virtualisation, Xen propose plusieurs options. Il propose deux ordonnanceurs : Borrowed Time virtuelle qui fournit un partage équitable des ressources CPU aux différents domaines, et Atropos qui est un ordonnanceur temps réel à contraintes relachées, il autorise un ordonnancement sur la base du best effort.

D'autres projets utilisent cette approche de paravirtualisation : Denali et Trango. Denali [13] partage le même principe général que Xen. Il faut que les systèmes d'exploitation soient adaptés au noyau Denali pour pouvoir être exécutés en tant que système invité. Trango [11] cible les set-top Box, les téléphones mobiles, etc. Il a été démontré que l'on pouvait sur un ARM9 utiliser deux instances Linux 2.6, une instance eCos 2.0 RTOS (Système d'exploitation temps réel), et un noyau de sécurité en même temps que des applications dans l'espace utilisateur.

Virtualisation assistée matériellement : Par rapport à l'approche précédente, la virtualisation assistée par le matériel permet d'utiliser un OS invité non modifié. La machine virtuelle dispose de son propre matériel virtuel et les OS peuvent être utilisés isolés les uns des autres. Dans de nombreux cas, la machine virtuelle exécute un système d'exploitation différent de celui de l'ordinateur hôte. Cette approche a été rendue récemment

possible depuis 2005/2006 grâce à une amélioration du jeu d'instructions des processeurs fournie par Intel (VT Technology) et AMD (Pacifica [2]), mais aussi par IBM (IBM advanced POWER virtualisation [6]) et Sun (Sun UltraSPARC T1 hypervisor [8]). L'objectif de cette démarche est de faire tourner les OS invités non modifiés sans l'inconvénient d'une diminution de la performance lors de l'utilisation de l'émulation ou de la paravirtualisation logicielle.

3 Virtualisation de l'accés au réseau

Dans cette section, nous analysons comment le mécanisme des machines virtuelles accéde et partage les cartes d'interface réseau physique. Dans un premier temps nous rappellons le principe d'implantation des protocoles dans le système d'exploitation et les interactions avec l'interface de communication matérielle. Nous détaillons ensuite les solutions originales pour permettre l'utilisation du réseau par les machines virtuelles proposées dans les diverses méthodes de virtualisation exposées précédemment.

3.1 Protocoles et accès réseau classiques

Dans cette section nous présentons l'approche utilisée dans Unix et plus spécifiquement sa version Linux pour l'implantation des protocoles et des drivers d'accès aux périphériques réseau.

Les applications communiquent à travers l'interface SOCKET qui permet l'accès aux services de transport (TCP, UDP, DCCP, SCTP...) implantés par les protocoles de transport du noyau. Ces protocoles interagissent avec les composants du protocole IP qui assure l'acheminement des paquets à travers les différents équipements intermédiaires jusqu'au destinataire. La couche IP dialogue avec la carte d'interface réseau (typiquement Ethernet) via son pilote (driver) situé dans le noyau et représenté par une "interface". A l'émission, le système copie les données dans une zone spéciale, le socket buffer (skbuf), puis rend la main à l'application. Pendant ce temps, le système d'exploitation se charge d'envoyer les données en arrière plan de cette zone vers la carte d'interface réseau par un accès direct à la mémoire (DMA, Direct Memory Access). Ainsi l'application n'est bloquée que pendant la première copie, au lieu d'attendre la réception du dernier message d'acquittement. En réception, la stratégie est symétrique, mais le récepteur est vraiment bloqué jusqu'à l'arrivée des données en mémoire utilisateur. Ce modèle a pour inconvénient d'être gourmand en temps processeur et en occupation du bus mémoire (3 accès côté émetteur et 3 côté récepteur). Ainsi, les communications à haut débit consomment un très grande quantité de temps CPU (les traitement protocolaire d'une connexion 10Gb/s saturent un ordinateur moderne). Le système d'exploitation fournit plusieurs fonctions pour le traitement de la pile protocolaire : l'ordonnancement du protocole, la gestion de la mémoire, la gestion des interruptions et les opérations d'accès au matériel. L'ordonnancement du protocole a une grande influence sur l'efficacité et les performances temps-réel du protocole.

Au niveau des performances que l'on peut attendre en terme de débit bout en bout entre les deux extrémités d'une connexion Ethernet, deux valeurs peuvent être considérées : le débit émis et le débit utile reçu. Soit R_{EthB} le débit binaire

Ethernet. $R_{\mathrm{EthB}} = 10^9 bps$ pour du Gigabit Ethernet. Nous définissons R_{Eth} le débit trame Ethernet en faisant l'hypothèse de trames de 1514 octets (sans le checksum (FCS), le préambule et l'*inter frame gap* (IFG)).

$R_{\mathrm{Eth}} = (1514/(1514 + 4 + 12 + 8)) * R_{\mathrm{EthB}} = 984.40\mathrm{Mbps}$

Nous definissons R_{IP} le débit IP paquets (Ethernet payload). Avec l'hypothèse d'une MTU[1] de 1500 octets : $R_{\mathrm{IP}} = (1500/1514) * R_{\mathrm{Eth}} = 975.30\mathrm{Mbps}$.

Le débit d'un protocole de transport idéal sur un lien 1Gbps Ethernet (sans *slow start* ni retransmissions) est défini par R_{TP}. Sous les hypothèses précédentes,

$R_{\mathrm{TP}} = ((1500 - 52)/1500) * R_{\mathrm{IP}} = 941.49\mathrm{Mbps}$

Ainsi cette valeur représente le *goodput* (R_{TP}) et (R_{EthB}) le débit émis effectivement sur le lien. Ainsi le débit utile maximum atteignable sur un lien gigabit Ethernet est donc de 941.49Mbps.

Au niveau de l'émetteur, le coût CPU C_{TPsnd} est modélisé comme suită :

$C_{\mathrm{TPsnd}} = C_{\mathrm{syscallsnd}} + C_{\mathrm{inetsnd}} + C_{\mathrm{TCPsnd}} + C_{\mathrm{IPsnd}} + C_{\mathrm{driversnd}}$

Au niveau du récepteur, le coût CPU C_{TPrcv} est modélisé comme suit :

$C_{\mathrm{TPrcv}} = C_{\mathrm{driverrcv}} + C_{\mathrm{IPrcv}} + C_{\mathrm{TCPrcv}} + C_{\mathrm{inetrcv}} + C_{\mathrm{syscallrcv}}$

Où C_{driver} correspond au coût de transmission tx ou de réception rx sur la carte (copie), C_{IP} au coût de traitement IP, C_{TCP} au coût de traitement TCP, C_{inet} au coût de copie dans/vers l'espace utilisateur, C_{syscall} le coût de l'appel système.

Les copies entre le système et l'espace utilisateur peuvent prendre jusqu'à 60% du coût CPU total de la transmission[**?**].

3.2 Protocoles et accès réseau dans les approches de virtualisation

Nous avons vu précédement que l'accès au réseau implique un certain nombre de composants du noyau (API socket, couche inet (interface générique pour le transport), couche transport (ici TCP), couche IP, driver Ethernet) et de composant matériels (carte d'interface, DMA, mémoire). Nous étudions ci-dessous comment la virtualisation vient se placer entre le noyau et le matériel dans les différentes approche.

VServer ne virtualise pas l'utilisation de l'interface réseau[4]. Il utilise seulement un alias IP différent pour chaque cas VServer. Cette approche peut créer des problèmes si différents VServers veulent utiliser la même adresse IP (par exemple 127.0.0.1). Si les opérations spéciales sont nécessaires, telles que NAT (Network Address Translation) ou de routage par exemple, elles devront se faire sur le serveur physique.

User Mode Linux offre deux principales possibilités[5] :

- Soit utiliser un démon hôte qui communique avec les différents machines virtuelles, par l'intermédiaire d'un socket Unix ;
- Ou utiliser un dispositif TUN / TAP (carte d'interface réseau virtuel) pour chaque machine virtuelle. L'hôte peut utiliser ces interfaces pour le routage, le mode passerelle, NAT ou quelque chose d'autre.

VmWare [12] utilise une carte réseau virtuelle AMD PCNET pour chaque machine virtuelle. Le système d'exploitation utilise un pilote approprié

[1] Message Transfer Unit

pour gérer cette carte. L'hôte utilise une technique similaire à Xen (voir ci-dessous) pour l'expédition du trafic réseau.

Xen [3] ne gère pas les périphériques directement, c'est le travail du noyau Linux exécuté en tant que *domaine 0*. Les drivers de toutes les cartes réseaux physique doivent être fournis par le noyau du domaine 0. Chaque VM possède une ou plusieurs interfaces réseau virtuelle. La communication entre les interfaces est effectuée en utilisant deux buffers circulaires à jetons (un pour l'envoi, un pour la réception). Xen fournit un nouveau module appelé *netloop* pour récupérer les paquets des interfaces réseau virtuelles des domaines Le domaine 0 peut utiliser un dispositif de routage, de pontage (bridge) ou de NAT virtuel pour transmettre les paquets sur les interfaces physiques.

3.3 Analyse de l'accès réseau dans XEN

Dans ce paragraphe nous proposons de modéliser l'accès réseau dans le cas de la virtualisation XEN.

Contrairement au cas sans la virtualisation, avec XEN, les paquets sont recopiés du socket buffer vers la zone mémoire associée au driver de la carte réseau du domaine 0. Pour gérer les multiples communication issues des différents domaines invités, XEN implante dans le domaine 0 un routeur, un pont ou même NAT. C'est cet équipement virtuel qui ordonnance les paquets et les transmet sur l'interface physique. Dans le cas du bridge virtuel, l'arbitrage entre les différentes communication se fait selon une simple politique FIFO, correspondant à l'ordre d'arrivée des paquets dans le driver du domaine 0.

La figure 2 schématise le principe du bridge logiciel proposé pour l'aiguillage des trames issues des différentes machines virtuelles.

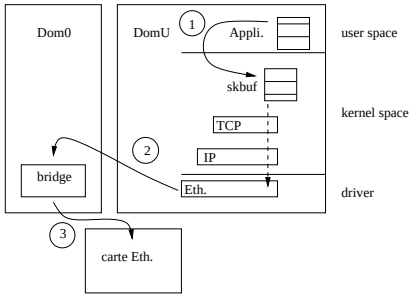


FIG. 1 – Le chemin des données et 3 copies avant d'atteindre le lien.

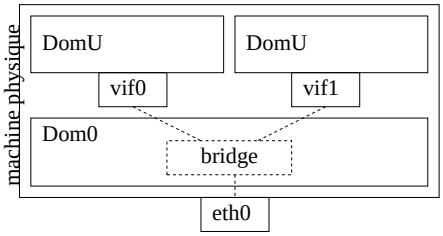


FIG. 2 – Bridge logiciel (dans le domaine 0) chargé de transmettre les trames Ethernet sur les cartes d'interfaces virtuelles des domaines invités.

Nous proposons le modèle suivant pour le coût CPU au niveau de l'émetteur. C_{TPsndV} est modélisé comme suită :

$C_{\text{TPsndV}} = C_{\text{syscallsnd}} + C_{\text{inetsnd}} + C_{\text{IPsnd}} + C_{\text{TCPsnd}} + C_{\text{netloopsnd}} + C_{\text{bridgesnd}} + C_{\text{driversndV}}$

Au niveau de l'émetteur, le coût CPU C_{TPrcv} est modélisé comme suit :

$C_{\text{TPrcvV}} = C_{\text{driverrcvV}} + C_{\text{bridgercv}} + C_{\text{netlooprcv}} + C_{\text{IPrcv}} + C_{\text{TCPrcv}} + C_{\text{inetrcv}} + C_{\text{syscallrcv}}$

Où C_{driverV} correspond au coût de transmission tx ou de réception rx sur la carte (copie 1) via le driver générique, C_{bridge} au coût de traitement du bridge, C_{netloop} au coût de copie (copie 2) du domaine invité vers le domaine 0 (ou réciproquement). C_{driverV} correspond au coût de la copie vers la carte d'interface. On peut s'attendre à ce que la copie entre le domaine invité et le domaine 0 constitue une part importante du coût CPU total de la transmission.

4 Expérimentations et analyse des résultats

L'objectif de ces expérimentations est d'étudier le surcoût de l'implantation de la virtualisation pour l'accès au réseau dans le cas de l'environnement XEN. Nous nous concentrons sur les performances de bout en bout obtenues au niveau utilisateur via le protocole TCP. Comme nous l'avons présenté dans la section précédente, la gestion des entrées-sorties réseau dans XEN fait intervenir un certain nombre de composants logiciels qui s'intercalent dans le chemin classique des données. Ces composants ne sont pas nécessairement activés séquentiellement et peuvent être arbitrairement invoqués par l'ordonnanceur des tâches. Par ailleurs, ces composants peuvent faire des copies supplémentaires de paquets, ce qui est très coûteux à haut débit. Nous avons donc développé un plan expérimental pour mesurer le coût dû à la virtualisation au niveau du client et au niveau du serveur en termes de débit et de capacité CPU (expérience 1). Le paramétrage de TCP étant relativement délicat dans le cas des réseaux haut débit, nous avons étudié les différentes valeurs de taille de buffer (expérience 2). Enfin, comme la virtualisation permet d'instancier plusieurs machines virtuelles sur une même machine, nous avons cherché à savoir si le partage des ressources était équitable et prévisible (expérience 3).

4.1 Plate-forme expérimentale et outils

Les résultats présentés dans cette section sont obtenus sur les équipements mis à disposition dans la plate-forme expérimentale Grid5000[1]. Il s'agit de serveurs IBM eServer 325 dotés de processeurs AMD Opteron dual-core 246 à 2GHz/1MB/400MHz et de 2GB de mémoire vive avec une interface GigaEthernet (pilote tg3). Ces machines physiques hébergent une, deux ou quatre machines virtuelles XEN. Chaque machine virtuelle exécute un systéme d'exploitation GNU/Linux et a une partition disque dédiée (système de fichiers non-partagés). Les machines physiques sont interconnectées directement via un commutateur Gigabit ethernet.

Afin de charger de manière contrôlée le processeur dans nos différentes expériences, nous avons développé un programme spécifique de simulation de charge utile, appellé le *softloader*. Il est constitué d'une boucle active effectuant des calculs en arithmétique entière, suivie d'une instruction de pause paramétrable (`nanosleep(long)`). Ainsi, à l'aide d'un ajustement approprié, nous pouvons charger le processeur à une valeur déterminée (ex : 50%, 90%...).

Pour générer et mesurer le trafic réseau nous avons utilisé l'outil de benchmark `iperf` [10] afin de générer des flux TCP depuis l'espace utilisateur des différentes machines virtuelles. Nous avons généralement utilisé le paramétrage par défaut de iperf. La taille de fenêtre de TCP n'a pas été modifiée directement mais via iperf.

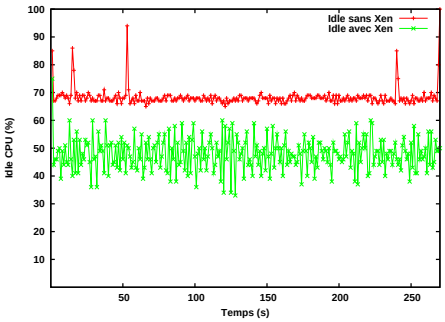


FIG. 3 – Ressource CPU disponible coté client (avec et sans XEN).

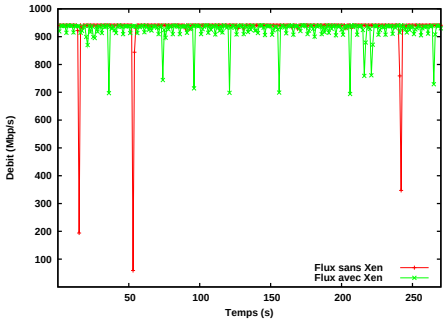


FIG. 4 – Débit réseau (avec et sans XEN)

Pour superviser la charge des processeurs (ou plus exactement des *cores*) nous avons utilisé `sar`. Ainsi cet utilitaire nous permet d'obtenir les mesures de références (sans virtualisation). Nous avons utilisé une version modifiée par nos soins de `xenmon` (programmé en Python) pour mesurer cette charge sous XEN.

4.2 Résultat de l'expérience 1 : coût pour une machine virtuelle

L'objectif de cette expérience est d'évaluer les performances de TCP dans le cas d'une machine utilisée uniquement pour la communication puis dans le cas d'une machine utilisée pour la communication et pour le calcul.

La figure 2 montre le coût brut de Xen (distance entre les courbes). Ces résultats sans charge CPU supplémentaire (le programme softloader n'est pas lancé).

Le débit réseau est très faiblement impacté par la présence de Xen. Il est de 938 Mbits/sec (941 Mbits/sec maxi constaté sur cette configuration). La figure 3 illustre ce modeste écart de débit, mais il met aussi en évidence des chutes de débit sporadique.

La figure 4 illustre le coût du transfert en terme de ressource CPU en mettant en évidence la charge laissée disponible. On observe ainsi que le CPU n'est utilisé qu'à 30 % dans le cas Linux standard. Le fait d'utiliser Xen entraîne un surcoût de 20 à 25% et les mesures mettent en évidences une instabilité dans la charge constatée. Nous avons constaté un résultat comparable au niveau du serveur avec un coût CPU de base légèrement plus important (35%).

Dans le cas d'une machine virtuelle exécutant simultanément une communication gigabit et un calcul saturant un coeur à 90 % nous observons que les performances réseau ne sont pas très affectées (Figure 7). La chute en débit est de quelques Mbps. Ces résultats encourageants sont néanmoins dû à la configuration bi-coeur des machines employées. Nous remarquons que la gestion des coeurs est faite par XEN et que les traitements protocolaires ne sont pas systématiquement faits sur le même processeur.

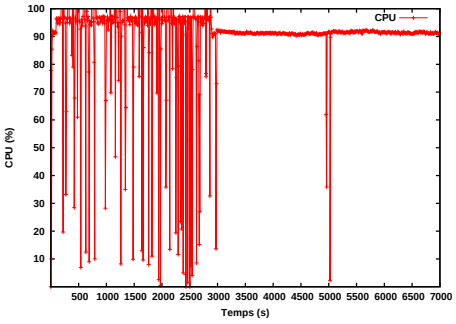


FIG. 5 – Ressource CPU disponible coté client en cas de charge à 90%

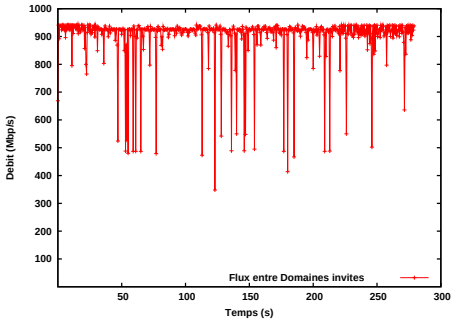


FIG. 6 – Performance TCP en cas de charge à 90%

4.3 Résultats de l'expérience 2 : taille des buffers

Dans le cas de l'utilisation de liens gigabit, il est connu que le paramétrage TCP (TCP tuning) a une grande importance. Aujourd'hui les valeurs optimales sont connues et les systèmes d'exploitation activent généralement l'autotuning. Dans le cas du transfert entre deux machines virtuelles situées sur une même machine physique la taille de fenêtre joue un rôle important. Dans cette série d'expérience, la taille des fenêtres TCP a été laissée par défaut. Cette taille est automatiquement calculée et optimisée par le noyau. Il reste à étudier si cette optimisation est efficace. On note par exemple que cette taille de fenêtre n'est pas la même pour le domaine 0 (64 Ko) que pour les domaines invités ou virtuels (16 Ko). Ce paramètre a de l'importance comme le montre ces mesures :

Communications Intra-machine
Domaine 0 vers Domaine virtuel (TCP Win 64Ko) : 610 Mbits/sec
Domaine virtuel vers Domaine 0 (TCP Win 16Ko) : 2.05 Gbits/sec

Par ailleurs, nous avons mené des expériences entre les domaine 0 et les domaines U entre 2 machines distinctes.

Machine Serveur			
Machine Cliente		Domaine 0	Domaine U
	Domaine 0	941Mbits/sec	925Mbits/sec
	Domaine U	941Mbits/sec	930Mbits/sec

Le débit entre deux domaine 0 est comparable à celui que l'on pourrait obtenir avec une configuration sans Xen. Les débits entre les domaines 0 et les domaines virtuels sont moins bons à cause de la mauvaise sélection de la taille des fenêtres TCP. La communication entre les domaines virtuels et les domaines 0 reste très bonne et ce malgrès une taille de fenêtre TCP de 16Ko. Il faut noter que dans le cas de cette expérience la latence est très faible ($< 1ms$). La communication inter domaines virtuels (la plus commune) présente de bonnes performances avec un surcoût de 10%.

4.4 Résultats de l'expérience 3 : Coût avec plusieurs machines virtuelles

L'objectif de cette expérience est d'étudier le comportement dans le cas de plusieurs machines virtuelles s'exécutant sur la même machine physique. Nous avons considéré ici uniquement des configurations symétriques : N machines virtuelles clientes sur 1 machine physique correspondant avec N machines virtuelles clientes sur 1 machine physique serveur. Nous avons fait varier N selon les valeurs 1, 2 et 4. Nous donnons les courbes correspondant au cas de 4 machines virtuelles avec uniquement des communications.

	1flux 1VM		2 flux 2 VM			4 flux 4 VM				
	TCP1	Total	TCP1	TCP2	Total	TCP1	TCP2	TCP3	TCP4	Total
A vide (Mb/s)	940	940	500	410	910	350	215	155	115	835
En charge (Mb/s)	938	938	290	290	580	60	50	60	60	230

FIG. 7 – Débits obtenus en fonction du nombre de machines virtuelles (en Mbits/s).

La table 4.4 donne l'évolution des débit avec et sans charge dans le cas de 1, 2 et 4 machines virtuelles. On observe un baisse quasi linéaire du débit aggrégé ainsi qu'un accroissement de l'inéquité interflux dans le cas des processeurs à vide. En cas de charge, les débits s'écroulent avec l'augmentation du nombre de machines virtuelles.

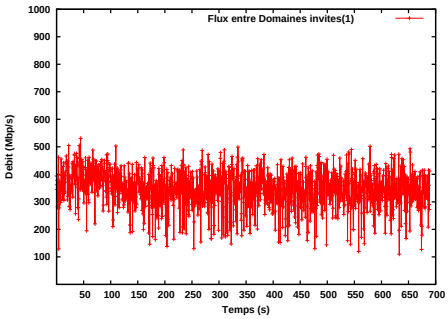


FIG. 8 – Flux TCP 1 - cas 4 domaines

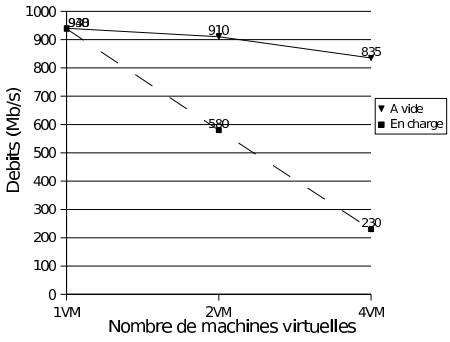


FIG. 9 – Débits réseau agrégés.

Les figures 10 et 11 explicite la répartition de la consommation des ressources dans les domaines 0 et U selon le nombre de machines virtuelles. On constate que cette consommation de ressources ne s'accroit pas linéairement. En particulier pour le domaine 0. En cas de charge, les communications semblent être fortement pénalisées au profit du calcul.

5 Travaux relatifs

La littérature dans le domaine de l'évaluation des performances réseau des systémes virtuelles reste encore très restreinte. Dans [7] une présentation de HP faite lors du XEN summit 4 montrent les résultats du "profilage" de la

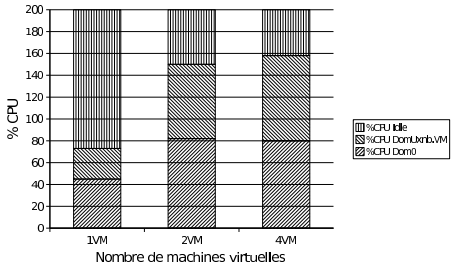


FIG. 10 – Répartitions du CPU lors de transferts (à vide).

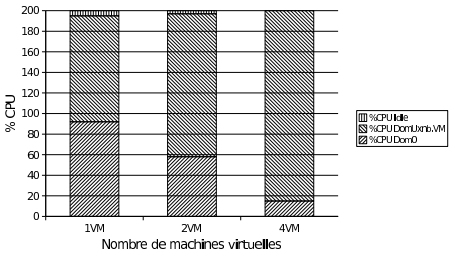


FIG. 11 – Répartitions du CPU lors de transferts réseau (en charge).

machine virtuelle XEN pour déterminer où se situe les surcoûts. Ces résultats sont complémentaires aux notres même s'ils montrent des performances sous-optimales pour les connexions gigabit, étant donné que l'obtention du gigabit est dépendant de nombreux paramètres matériels, il semble que leur configuration soit limitée.

6 Conclusions et travaux futurs

Les résultats obtenus montrent que le débit d'une connexion seule sur un lien Gigabit n'est quasiment pas affecté par la virtualisation même dans le cas d'une machine chargée à 90%. En contre partie, bien que ces débits réseaux sont obtenus au prix d'une consommation CPU supérieure et d'un ajustement correct des tailles de buffers. Lorsque l'on multiplie le nombre de machines virtuelles le surcoût dû à la virtualisation augmente linéairement avec le nombre de machines en terme de performance réseau. On observe un accroissement important de l'inéquité entre flux ce qui pose des problèmes dans le cas de l'utilisation de la virtualisation dans les routeurs.

En termes de travaux futurs, nous nous nous proposons de mener une étude détaillée conjointe de l'ordonnancement des flux au niveau du domaine 0 et de l'ordonnancement des tâches dans XEN. Par ailleurs, les machines que nous avons utilisé pour nos expérimentations ne proposent pas le support de la virtualisation matérielle. Il serait interressant de refaire les mêmes mesures sur des machines offrant le support matériel de la virtualisation. A plus court terme, notre prochain objectif est d'évaluer le surcoût de la virtualisation sur le délai et la jigue. Enfin, à plus long terme nous souhaitons tirer avantage des cartes d'interfaces réseau programmables (équipés de *Network Processor*) en y déportant le switch virtuel ce qui devrait soulager le ou les processeurs physiques.

Références

[1] Grid5000. https://www.grid5000.fr/.

[2] AMD. Pacifica x86 virtualization. http://enterprise.amd.com/us-en/AMD-Business/Business-Solutions/Consolidation/Virtualization.aspx.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03 : Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM Press, 2003.

[4] J. Gelinas. Virtual private servers and security contexts. In *http ://www.solucorp.qc.ca/miscprj/s_context.hc*, 2003.

[5] Hans-Jorg Hoxer, Kerstin Buchacker, and Volkmar Sieh. Implementing a user mode linux with minimal changes from original kernel. In *Proceedings of 9th International Linux System Technology Conference*, Cologne, Germany, September 2002.

[6] IBM. Ibm advanced power virtualization. http://www-03.ibm.com/systems/p/apv/.

[7] Aravind Menon, Willy Zwaenepoel, Jose Renato Santos, Yoshio Turner, and John Janakiraman. Diagnosing performance overheads in the xen virtual machine environment. *Xen Summit 4, Yoktown, NY, USA*, april 2007.

[8] Sun Microsystems. Sun ultrasparc t1 hypervisor. http://opensparc-t1.sunsource.net/specs/Hypervisor-api-current-draft.pdf.

[9] Ian Pratt and Keir Fraser. Xen 3.0 and the art of virtualization. *Linux Symposium*, 2005.

[10] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. iperf : testing the limits of your network. http://dast.nlanr.net/Projects/Iperf/.

[11] Trango. http://www.trango-vp.com/.

[12] Carl A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI) :181–194, 2002.

[13] Andrew Whitaker, Marianne Shaw, and Steven D. Gribble. Scale and performance in the denali isolation kernel. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI 2002), Boston, MA*, December 2002.