Mini Hyperviseur avec KVM

Labo de virtualisation des systèmes réalisé par Quentin Rod et Thomas Dagier durant le 6ème semestre ISC à HEPIA.

1. Etat du projet

À ce stade du projet, tous les points énnoncés dans ce TP ont été correctement implémentés. Nous n'avons, à notre connaissance, pas identifié de bug sur les différentes fonctionnalités ajoutées.

2. Conception

À travers ce projet, nous avons implémenter une émulation et une paravirtualisation d'un affichage VGA sur 25 lignes, 80 colonnes et 16 couleurs. De même pour l'écriture d'un disque IDE avec protocole ATA.

2.1. Affichage VGA

- Affichage VGA avec périphérique émulé:

Il est nécessaire d'avoir une mémoire partagée en Read Only (MMIO) entre la VM et l'hyperviseur. Elle est mappée à l'adresse 0xB8000 de la VM afin qu'elle puisse y écrire des caractères VGA comme si elle accédait le matériel. À chaque écriture dans la mémoire, une exception est générée et l'hyperviseur récupère le caractère VGA à afficher. Il doit ensuite le faire apparaitre à l'écran.

- Affichage VGA avec périphérique paravirtualisé:

Contrairement à l'émulation, une mémoire accessible en lecture et écriture est partagée entre la VM et l'hyperviseur. La VM sait qu'elle est paravairtualisée et à connaissance de l'adresse de cette zone mémoire. Elle peut donc inscrire la donnée à afficher ainsi que ces coordonnées . Elle précise ensuite le numéro d'hypercall correspondant sur le port `0xABBA` . À chaque écriture sur un port PMIO, une exception est générée et l'hyperviseur récupère la donnée inscrite pour l'afficher.

2.2. Ecriture sur le disk

- Ecriture sur le disk avec périphérique émulé:

La VM ne sachant pas qu'elle tourne sur un hyperviseur, elle va vouloir écrire sur le matériel à travers les registres PMIO. Cela génère une interruption permettant au VMM de récupérer la valeur. Il doit alors répondre à la VM comme si il s'agissait du matériel.

Lorsque les données reçues font la taille d'un secteur et que la VM a effectué correctement la procédure, le VMM va écrire les données sur l'image disque réservé à la VM.

- Ecriture sur le disk avec périphérique paravirtualisé:

Similairement au VGA, il y a une mémoire accessible en lecture et écriture partagée entre la VM et l'hyperviseur. La VM. ayant connaissance de l'adressse de la zone mémoire, elle peut y inscrire les données à écrire sur le disque aisni que l'adresse du secteur. Elle doit également indiquer le numéro d'hypercall correspondant sur le port `0xABBA` . Lors de l'exception proovquée par l'écriture, le VMM écrit ces données sur l'image disque correspondant à la VM.

3. Réalisation

Afin d'alléger le code de l'hyperviseur, la gestion du VGA et du disque a été placée respectivement dans "vga_vmm", "disk_vmm". Le fichier "hypercall" contient les informations liées aux hypercalls comme les numéros disponibles ou encore les types de données à utiliser.

3.1. Affichage VGA

- Affichage VGA avec périphérique émulé:

La mémoire partagée (MMIO) allouée au moment de la création de la VM est alignée à une page de 4KB et est multiple de 4KB.

Chaque caractère VGA reçu est stocké dans une structure de données qui facilite l'affichage des caractères par le VMM. Un tableau de cette structure représentant l'écran de 25 par 80 est donc présent côté hyperviseur. Les caractères sont stockés dans ce tableau par ordre de réception, de gauche à droite et de haut en bas.

Ce tableau est affiché à l'aide d'un thread qui gère une fenêtre SDL avec un taux de rafraichissement à 60Hz. Ce dernier est réalisé par une mise en pause du thread pendant la durée correspondante. Pour afficher le texte, le VMM utilise la police fournie. Il fait également apparaitre leur couleur de texte et celle de fond. La fenêtre SDL peut être fermée avec la touche `escape` . La librarie met en effet à disposition une fonction pour connaitre la touche appuyée.

La fermeture de la fenêtre indique une fin d'utilisation de la VM. Le vCPU étant en attente d'un VM_EXIT, il ne peut pas s'arrêter. Ce dernier est exécuté dans un thread pouvant être annulé de manière asynchrone. Suite à la destruction de la fenêtre, une annulation est effectuée par le VMM.

- Affichage VGA avec périphérique paravirtualisé:

En mode paravirtualisé, la VM n'écrit plus sur le port 0xB8000 mais insère directement le caractère dans la mémoire partagée entre la VM et l'hyperviseur. Cette dernière est allouée au moment de la création de la VM et est alignée à une page de 4KB. Elle est aussi multiple de 4KB. Uniquement quelques octets sont utilisés pour stocker le caractère et ses couleurs.

Suite à l'écriture des informations du caractère dans la mémoire, la VM notifie l'hyperviseur d'une écriture VGA en indiquant l'index `VGA=1` sur le port `0xABBA` . L'hyperviseur peut donc effectuer la fonction correspondant à cette hypercall. Cette dernière consiste à insérer les données du caractères dans le tableau local à l'indice correspondant aux coordonnées du pixel. Cêst ce tableau qui est affiché dans la fenêtre SDL. Afin de permettre plus de flexibilité dans l'implémentation des hypercalls, un tableau de fonctions est utilisé. En fonction de l'index écrit par la VM sur le port `0xABBA` , le VMM appelera la fonction relative à l'affichage VGA, à l'écriture sur le disk ou n'importe quel autre hypercall que l'on a ajouté.

3.2. Ecriture sur le disque

- Ecriture sur le disque avec périphérique émulé:

Une machine d'état est mise en place afin de s'assurer que la procédure soit respectée. Dès lors qu'une étape est mal réalisée par la VM, le VMM retourne à l'état initial. En revanche, si elle est correcte, les données sont écrites au secteur souhaité sur l'image disque donnée par l'utilisateur.

- Ecriture sur le disque avec périphérique paravirtualisé:

Il n'y a plus de procédure nécessaire pour écrire sur le disque. La VM indique dans la mémoire partagée, les 512 octets ainsi que l'adresse du secteur. Lors de l'interruption, le VMM écrit directement dans l'image disque au secteur voulu.

4. Résultats

La paravirtualisation du VGA proposée dans ce TP n'est pas optimisée. Il est nécessaire d'effectuer un hypercall pour chaque caractère écrit par la VM. Il serait intéressant d'écrire toute la fenêtre dans la mémoire partagée puis de faire un unique hypercall. D'autant plus que la taille d'une page suffit largement pour écrire tout le framebuffer et que beaucoup de fragmentation interne est présente. Avec la méthode actuelle, les performances en émulation et en paravirtualisé sont proches.

En revanche, pour le disque, la suppression de la machine d'état accélère le processus. La mesure de performance consiste à comparer le temps d'écriture entre l'émulation et la paravirtualisation. Pour ceci, nous écrivons un même nombre de secteurs. Ces fonctions sont présentes dans les fichiers benchmark_disk_emul et benchmark_disk_pv. Deux règles nommées benchmark_disk_pv et benchmark_disk_emul sont également ajoutées au makefile. Elles permettent de générer le binaire, l'exécuter avec KVM et mesurer le temps d'exécution avec la fonction time.

nombre de secteurs	emulation	paravirtualisation
100	0.061s	0.005s
1'000	0.583s	0.015s
10'000	5.164s	0.06s
100'000	54.815s	0.6s
1'000'000	7min26s	3.8s
5'000'000	31min12s	19.1s

Avec 5'000'000 écritures le gain de temps est conséquent avec la paravirtualisation. L'exécution est 98 fois plus rapide.

5. Conclusion

Ce travail nous a permis de comprendre comment fonctionne une émulation et une paravirtualisation avec KVM. À travers ce projet, nous avons rencontré des difficultés pour l'émulation du VGA. Notamment pour récupérer des données de tailles différentes et utiliser la police d'écriture dans la fenêtre SDL.

Nous nous sommes logiquement rendu compte qu'implémenter un périphérique émulé demande beaucoup plus de temps et de réfléxion que de le faire avec la paravirtualisation. C'est la raison pour laquelle nous n'avons pas eu de soucis pour effectuer paravirtualisation des périphériques. D'autant plus que nous avions déjà fait le travail pour l'émulation donc nous savions quoi faire.