BREF RESUME DES LECTURES EFFECTUEES SUR LE THEME

PML

Page Modification Logging

N’DONGA BITCHEBE CELESTINE STELLA

1 INTRODUCTION

WORKING SET SIZE 2

3 WORKING SET SIZE ESTIMATION

BADIS 4

5 CONCLUSION

INTRODUCTION

CONTEXTE

De plus en plus d'entreprises et de particuliers utilisent les services de cloud pour conserver des documents importants ou externaliser au mieux leurs services. Les utilisateurs du cloud cherchent à s'assurer de garder le même rendement tant sur le plan de la confidentialité (données), que des performances (applications). C'est la raison pour laquelle les techniques de virtualisation traditionnelles ne répondent plus face à l'affluence des utilisateurs sur les plateformes de cloud.

Devant cette demande accrue, plusieurs labos tournent leurs recherches vers l'amélioration des techniques de virtualisation. Intel, à l'occurrence, s'atèle à améliorer l'utilisation de la mémoire pour les gestionnaires de machines virtuelles. C'est ainsi qu'on arrive au PML (Page Modification Logging), qui est une amélioration de la technologie de virtualisation d'Intel (Intel VT) qui étend les capacités d'un moniteur de machine virtuelle, afin que ce dernier puisse gérer de façon efficace la mémoire lors de l'exécution de la machine.

INTRODUCTION

PROBLEMATIQUE

L'enjeu ici est d'implémenter des algorithmes de gestion mémoire et de classification de page mémoire, afin d'optimiser des opérations telles que la défragmentation ou la pagination lors de l'exécution de la machine virtuelle.

Pour cela, il faudrait analyser le gestionnaire de machine virtuelle et collecter des statistiques lors de son exécution.

Le problème est que ces analyses imposent des coûts de latence et de bande passante qui pourraient avoir des impacts inacceptables pour les performances de la machine.

LE MODELE « WORKING SET » (WS)

DEFINITION

Un processus étant segmenté en mémoire sous forme d'unités appelées « pages mémoire », il peut arriver que lors de l'exécution, certaines pages du processus soient manquantes : on parle alors de « défaut de page ». Lorsque le gestionnaire de la mémoire centrale est confronté à un défaut de page, le processeur doit effectuer des « swap » pour aller chercher les pages manquantes, ce qui constitue des opérations coûteuses et entraine des temps de latence dans l'exécution. S'il était donc possible de connaître à tout moment quelles pages seraient utilisées par le processus on aurait une meilleure exploitation du temps et des ressources mémoire et processeur.

Le modèle ici présenté s'atèle donc à définir une collection minimale de pages (pages mémoire) devant être chargées en mémoire centrale pour qu'un processus fonctionne de façon efficace, sans défaut de page : le working set.

LE MODELE « WORKING SET » (WS)

ALGORITHMES

Les défauts de page entrainent un important « traffic de pages », définit comme le nombre de pages déplacées entre les mémoires (centrale et auxilliaire) par unité de temps.

Pour minimiser le traffic de pages, il existe certains algorithmes qui permettent de sélectionner, suivant certains critères, les pages qui resteront en mémoire centrale :

Random Selection

On choisit de façon aléatoire les pages à remplacer en mémoire

⇩

Certaines pages fréquemment utilisées sont retirées ce qui conduit à un traffic plus important

FIFO Selection (First In First Out)

Ici on choisit la page la moins récemment paginée

⇩

- Efficace uniquement pour les programmes séquentiels
- Pour les autres, cela ne fera qu'augmenter le nombres de défauts de pages

LE MODELE « WORKING SET » (WS)

ALGORITHMES

LRU Selection (Least Recently Used)

Pour libérer de la place en mémoire centrale, on retire de la table des pages, celles qui n'ont pas été référencées depuis longtemps

⇩

Lorsque plusieurs processus réclament de la mémoire, cette méthode peut conduire à une surcharge du processeur

ATLAS Loop Detection Method

En détectant les comportements de boucles et cycles dans les processus, on retire de la mémoire les pages susceptibles de ne pas être utilisées avant longtemps

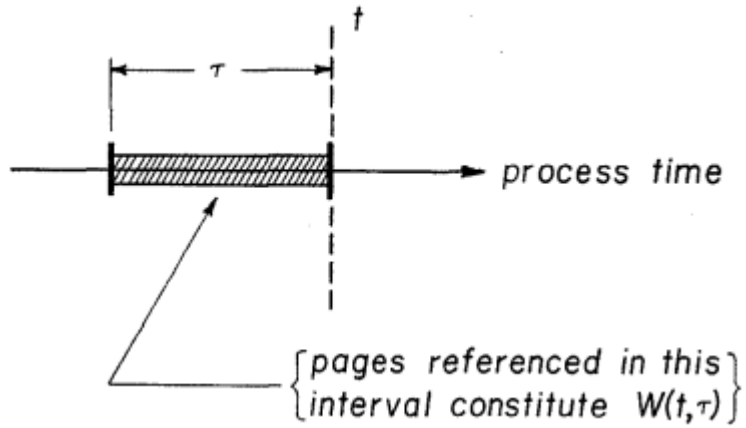
⇩

- Efficace uniquement pour les programmes cycliques
- Implémentation coûteuse

LE MODELE « WORKING SET » (WS)

CARACTERISTIQUES

Un modèle mathématique a été développé autour de ce modèle, et du point de vue du système, le WS ou « jeu d'information » associé à un processus à l'instant t, noté $\boldsymbol{W(t,\tau)}$, est l'information référencée par ce processus durant l'intervalle $\boldsymbol{(t-\tau,\tau)}$.



Un WS a quatre principales propriétés :

La taille

Elle est notée $\boldsymbol{\omega(t,\tau)}$ et vu qu'aucune page n'est référencée au début on a : $\boldsymbol{\omega(t,\tau)=0}$

La prédiction

Le comportement de référence la dernière page immédiate est un bon prédicteur du comportement de référence pour la prochaine page immédiate. Pour un faible écart de temps $\boldsymbol{\alpha}$, tel que $\boldsymbol{\alpha<\tau}$, la prédiction se traduit mathématiquement par :

$$\boldsymbol{W(t+\alpha,\tau)=W(t+\alpha,\alpha)\cup W(t,\tau-\alpha)}$$

LE MODELE « WORKING SET » (WS)

CARACTERISTIQUES

"Le taux de rentrée"

C'est le temps moyen de traitement au bout duquel une page retourne dans le WS. Il est noté $\lambda(\tau)$

La " τ-sensibilité"

Il permet de mesurer comment varie le taux de rentrée en fonction de τ

De toutes ces caractéristiques, la plus importante est la "taille" du WS, en abrégé WSS (**Working Set Size**).

En effet, elle permet d'anticiper l'évolution de la mémoire du WS des processus afin de pouvoir ajuster de façon dynamique la mémoire allouée à la machine virtuelle.

ESTIMATION DU WORKING SET SIZE (WSS)

DEFINITION ET PROBLEMATIQUE

Comme mentionné au début, le but est d'optimiser l'utilisation de la mémoire. Or, lorsqu'on alloue de façon statique de la mémoire à une machine virtuelle, elle n'est pas totalement utilisée à tout instant. Ils y a des périodes d'exécution où les processus n'ont besoin souvent que de 10% de la mémoire totale dont ils disposent : il est donc intéressant de pouvoir allouer et désallouer l'espace mémoire inutilisé ou nécessaire lorsque le besoin se pose. Ainsi, en estimant la taille du WS des processus nous pouvons rentabiliser la mémoire et la gérer de façon plus efficace.

Le problème peut se résumer en trois questions principales :

Q1) Comment obtenir l'activité sur la mémoire de la machine virtuelle, sachant que celle-ci est une boîte noire?

Q2) Comment estimer le WSS de la machine virtuelle à partir des données collectées?

Q3) Comment mettre à jour les capacités de la machine pendant son exécution?

ESTIMATION DU WORKING SET SIZE (WSS)

TECHNIQUES

Il existe plusieurs techniques qui permettent d'estimer le WSS. La technique optimale doit répondre aux questions Q1 et Q2 énoncées plus haut tout en présentant les propriétés suivantes :

- Être non intrusive (ne nécessite pas la modification de la machine virtuelle)
- Être inactive (n'altère pas le cours d'exécution de la machine)
- Être précise
- Ne pas provoquer de surcharge (ni de la machine ni de l'hyperviseur)

"Committed_AS"

R1) – C'est le système d'exploitation qui gère tous les appels d'allocation de mémoire (donc il est possible de connaître l'activité mémoire de la machine virtuelle)

R2) – le système d'exploitation agrège la mémoire allouée à tous les processus pour estimer le WSS de la machine

ESTIMATION DU WORKING SET SIZE (WSS)

TECHNIQUES

”Zballoond”

R1) – l’idée basique derrière cette technique consiste à décroître graduellement la taille de la mémoire tant les compteurs sont positifs

R2) – ainsi, le WSS de la machine est considérée comme la plus petite taille qui conduit la machine à aucun swap ni défaut de page

”VMware”

R1) – l’hyperviseur sélectionne périodiquement et aléatoirement n pages de la mémoire de la machine et les invalide. Ainsi, lors du prochain accès à ces pages, il pourra compter le nombre f de pages, parmi celles invalidées, qui sont sujettes à un défaut de page

R2) – dans ce contexte, le WSS de la machine est estimée par la formule $\frac{f}{n} m_{cur}$ où m_{cur} est la taille actuelle de la mémoire (avant invalidation des pages)

”Geiger”

R1) – Geiger observe le traffic de page entre la mémoire tampon du système d’exploitation hôte et le swap

R2) – pour répondre à Q2, Geiger se base sur la technique de la ”mémoire fantôme” (m_{ghost}). Cette dernière représente une mémoire imaginaire qui étend la mémoire physique de la machine virtuelle. Il estime donc le WSS par la formule $m_{cur} + m_{ghost}\ si\ m_{ghost} > 0$

ESTIMATION DU WORKING SET SIZE (WSS)

TECHNIQUES

”Hypervisor Exclusive Cache (HEC)”

Cette technique est similaire à celle de Geiger. Ici, chaque machine virtuelle a une quantité de mémoire dite directe, le reste étant gérée par l’hyperviseur comme mémoire exclusive. C’est cette mémoire exclusive qui est assimilée à la mémoire fantôme de Geiger

”Dynamic Memory Pressure Aware Ballooning (DMPAB)”

R1) – cette technique introduit un ensemble d’hypercall à travers lesquels chaque machine virtuelle reporte à l’hyperviseur son nombre de pages (pages actives, inactives, etc.)

R2) – on définit ici trois états possibles de pression de mémoire : bas (la somme des pages de toutes les machines virtuelles est inférieure à la mémoire totale de l’hôte physique), moyen(les mémoires sont égales) et élevé.

”Machine learning”

R1) – cette technique s’appuie sur les données récoltées à partir des capteurs de performance du matériel

R2) – l’estimation se fait par régression multivariée sur les données collectées

ESTIMATION DU WORKING SET SIZE (WSS)

TECHNIQUES

Il faut maintenant évaluer pour chacune des techniques présentées plus haut, les métriques que doit respecter une technique optimale : l'intrusion, l'activité, la précision et la surcharge.

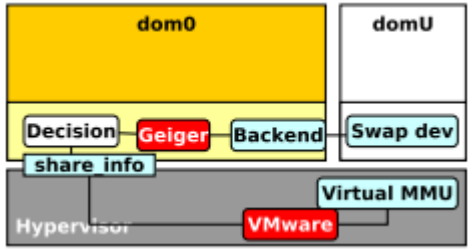
Technique	Intrusive	Active	Précise	Surcharge de la machine virtuelle	Surcharge de l'hyperviseur
Committed_As	Oui	Non	Non	Non	Non
Zballoond	Oui	Oui	Non	Oui	Non
VMware	Non	Oui	Non	Non	Oui
Geiger	Non	Non	Oui seulement si $m_{ghost} > 0$	Non	Non
HEC	Non	Oui	Oui si le cache exclusif est supérieur à 0	Non	Non
DMPAB	Oui	Oui	Oui pour les datacenter privés	Oui	Oui
Machine learning	Non	Non	Dépend de la quantité de données	Non	Non

BADIS

PRESENTATION

La plupart des solutions présentées sont souvent désavouées à cause de leur intrusion dans la machine virtuelle. Badis est une solution qui combine les techniques existantes de manière à répondre au problème sans intrusion dans la machine et en minimisant l'impact sur les performances.

Badis jumelle dans son architecture les techniques Vmware et Geiger comme présenté sur le schéma ci-après :



La technique Vmware est implémentée au niveau de l'hyperviseur tandis que Geiger est localisée à l'intérieur du dom0.

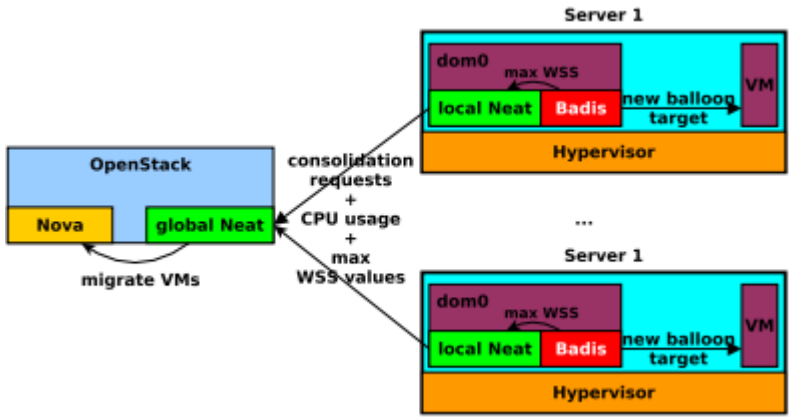
BADIS

CLOUD ET VIRTUALISATION

Dans le domaine du cloud, la consolidation des serveurs permet de regrouper les ressources des serveurs afin d'optimiser leur utilisation.

En général, c'est la mémoire qui limite l'utilisation des serveurs consolidés : en effet dans la plupart des systèmes de virtualisation, la mémoire réservée est entièrement allouée au lancement de la machine virtuelle; cette quantité devrait atteindre la mémoire totale demandée par la machine lors de son exécution; mais dans la plupart des cas les applications déployées n'utilisent pas plus de 50% de la mémoire allouée à la machine virtuelle sur laquelle elles sont déployées. Badis cherche donc à estimer la taille du Working Set des processus afin d'optimiser au mieux l'utilisation des serveurs dans les systèmes de virtualisation et d'augmenter ainsi le ration de consolidation.

Badis est basé sur le système de consolidation OpenStack Neat selon l'architecture suivante



CONCLUSION

Ce résumé présente l'essence du PML (Page Modification Logging), amélioration de la technologie de virtualisation d'Intel (Intel VT). Cette technologie s'intéresse de façon globale à l'optimisation de l'utilisation de la mémoire par les serveurs dans les systèmes de virtualisation.

Plusieurs techniques et algorithmes existent déjà qui traitent du problème, mais sont limitées soit par leur intrusion dans les machines virtuelles soit par les dégradation de performance qu'elles peuvent entrainer sur les applications.

Pour connaître la quantité approximative de mémoire qu'il faudrait allouer aux machines pour éviter les "pertes" de mémoire (mémoire allouée mais non utilisée), un modèle a été pensé, celui du « Working Set », défini comme la collection minimale de pages (pages mémoire) devant être chargées en mémoire centrale pour qu'un processus fonctionne de façon efficace, sans défaut de page. Ainsi, il suffit d'estimer la taille de ce working set pour connaître la quantité de mémoire à allouer. Ce sur plan, la technique Badis combine plusieurs autres techniques existantes pour avoir une solution non intrusive, qui ne modifie pas le flow d'exécution des machines et qui permet d'avoir une estimation à précision acceptable. Badis est basé sur le système OpenStack Neat.