Vol.2012-DPS-152 No.32
Vol.2012-GN-85 No.32
Vol.2012-EIP-57 No.32
2012/9/14

Linux仮想化環境におけるメモリコミットの分析

中尾 司ピエール[1,a)]　坂下 善彦[2]

概要：Kernel-based Virtual Machine(KVM) により構築した仮想化環境においてホスト OS の Linux カーネルによって提供される Kernel Samepage Merging(KSM) を利用することによりホスト OS のメモリが節約されることが分かった。特に KSM はゲスト OS が多く存在するシステムにおいて有効である。　本研究では，KSM を利用することによりマージされるメモリ容量に注目し， VM に搭載されているゲスト OS の仮想メモリ管理における実メモリの実行制御が，ホスト側の OS の実メモリ管理の箇所で実施される状況を観測し，マージされるメモリ容量がシステム全体に寄与する効果を観測した。

キーワード：キーワード：仮想マシン, 資源管理, メモリ管理

An observation of the memory commitment on Linux Virtual Machine Environments

Nakao Tsukasapierre[1,a)]　Sakashita Yoshihiko[2]

Abstract: In a virtual environment was constructed by (KVM), that the memory of the host side will be saved by taking advantage of (KSM) are expected to Kernel Samepage Merging be provided by the Linux kernel on the host OS Kernel-based Virtual Machine(VM). In particular, KSM is effective in this system there are many guest OS. In this study, we focused on the amount of memory that can be merged by taking advantage of KSM. Control the management of real memory in guest OS virtual memory execution is installed in the VM, is carried out in place of real memory management of OS on the host side. We observed the effect of the amount of memory that can be merged to contribute to the overall system.

Keywords: Virtual machine, virtual memory, memory management, resource management

1. 背景・目的

昨今普及の一途をたどるクラウドコンピューティングにおいて Infrastructure as a Service(以下 IaaS) と呼称されるものがある。IaaS は仮想化されたコンピュータ資源をインターネット経由でサービスとして提供するものである。IaaS がコンピュータ資源のサービスを提供する方法の一つとして仮想機械または仮想マシン、バーチャルマシンと呼ばれるコンピュータの動作やソフトウェアの動作をエミュレートするソフトウェアが存在する。これは IaaS を構成する上での一つの要素でもある。仮想機械は主にシステム仮想機械と呼ばれるものが使われている。システム仮想機械は CPU 資源、メモリ資源、ハードディスク資源、その他入出力装置などシステム全体を再現することによりその上で OS を動かすことを可能とするものである。システム仮想機械で複数の仮想機械の資源を効率的に配分するプログラムをハイパーバイザと呼ぶ。システム仮想機械の上で OS を動かすことでそれが IaaS においてのコンピュータ資源となり提供される。また、IaaS はその構成と用途から分散システムの一つのあり方としても考えられる。以上を踏まえ我々は、仮想機械を用いた分散システムにおける資源の利用状態 (状況を) の把握をする上でメモリ資源に注目し

[1] 湘南工科大学 工学研究科　博士前期課程　電気情報工学専攻
Shonan institute of technology Graduate school of electric and information science
[2] 湘南工科大学 工学部 情報工学科
Shonan institute of technology Department of information science
[a)] 11t2013@sit.shonan-it.ac.jp

Vol.2012-DPS-152 No.32
Vol.2012-GN-85 No.32
Vol.2012-EIP-57 No.32
2012/9/14

た。仮想機械は資源を利用する上で仮想機械を実現する側 (ホスト) のハードウェアをハイパーバイザが効率よく分配することにより実現される。仮想機械は単一ホスト上で複数が稼働している状況で利用されており複数の仮想機械が共通の資源を利用している。

各仮想機械が要求する資源を割り当てる上で必要なメモリは静的に割り当てられる。ホスト側の管理者としては要望の一つとしてメモリ資源の工夫が考えられる。メモリ資源の工夫次第では一つのホスト上で更に複数の仮想機械を稼働させることが可能であるからである。これはハードウェアの資源を有効利用するために重要である。

本研究は仮想化環境を構築し仮想機械のメモリ資源の利用状況について検証、分析を試みる。仮想化環境は Linux カーネル仮想化基盤である Kernel-based Virtual Machine (以下 KVM) にて構築し、KVM 上で稼働する仮想機械を検証対象としている。Linux 環境下では Kernel SamePage Merging(以下 KSM) という機構がカーネルに存在する。ホスト側カーネルにて KSM を有効するとホスト側のメモリが節約される機構である。以上のメカニズムをもとにメモリが節約される様子を取得し、更に複数の仮想機械を稼働させる環境においての性能分析を行う。2 章以降は検証にあたっての背景技術について述べる。

2. Linux Kernel-based Virtual Machine

Linux KVM は Linux カーネル 2.6.20 から標準搭載されている仮想化基盤である。Linux KVM は Linux カーネルモジュールとして実装されている。Intel VT もしくは AMD-V を使ったネイティブ仮想化をサポートしている。ネイティブ仮想化とは OS を修正することなく隔離された状態で動作させることである。Linux KVM は周辺装置のエミュレーションは行わずそれらはデバイスエミュレータである qemu-kvm によって行われる。Linux KVM の特徴は Linux カーネルの一部となって動作するため Linux カーネルの機能を利用することができる。特にメモリ管理、プロセススケジューリングの機能などをそのまま利用しているため実装はコンパクトである。KVM 上で動作する仮想機械はホスト OS 側からは一つのプロセスとして見える。

2.1 仮想化を構成するソフトウェア

Linux KVM は複数のプログラムが連携することによって仮想機械を実現している。Linux カーネル本体に加え Linux KVM のカーネルモジュール、ユーザーモードで動作する qemu-kvm によって構成されている。のカーネルモジュールはユーザーモードのプロセス要求に応じて、仮想マシンにメモリを割り当て実行する機能を持っている。qemu-kvm はユーザーモードプロセスとして動作する。前述のとおり I/O デバイスを再現するためのものである。図 1 に KVM による仮想化の構成を示す。

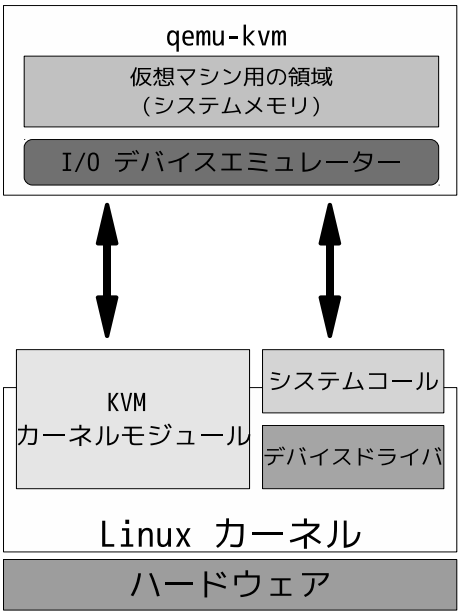

図 1　仮想化の構成

3. Kernel SamePage Merging の概要

Kernel SamePage Merging(以下 KSM) は Redhat 社によって提案、実装 [1] されたメカニズムである。Linux カーネル 2.6.32 でマージされた。KSM はユーザプロセスの匿名ページ領域をスキャンし、同一内容のページが存在した場合一つのページへマージすることでメモリ使用量の低減を実現する Linux カーネルの機能である。スキャンの際には ksmd というカーネルスレッドが指定間隔に合わせて指定ページ数をスキャンする。スキャンしたページを同定するために memcmp が用いられている。マージされたページは CoW(Copy on Write) 状態にされる。CoW の動作は Linux カーネルにある機能をそのまま利用している。マージされているページは現状の実装ではスワップアウトされない。マージされているページに更新が発生場合、マージされていたページはカーネルによって自動でメモリーページの複製を作成する。マージされているページの更新確認にはハッシュ値が用いられている。 ハッシュ値は jhash2 によって求められている。KSM のプログラマ向けインタフェースとして madvise(2) が存在する。スキャン対象となるメモリ領域は madvise に対して MADV_MERGEBLE を引数として与えることによってスキャン対象となる。KSM は/sys/kernel/mm/ksm にあるファイルで設定とデータの収集が可能となっている。/sys/kernel/mm/ksm 以下のファイルについて表 1 に示す。

KSM はページの探索、挿入、削除のアルゴリズムに赤黒木を用いている。赤黒木は O(log n) 以下のオーダーで動作する (n は KSM によってスキャンされたページの合計である)。このため短時間に探索、挿入、削除が実現できている。KSM を有効に使用する推奨条件として、複数の同

Vol.2012-DPS-152 No.32
Vol.2012-GN-85 No.32
Vol.2012-EIP-57 No.32
2012/9/14

表 1　KSM の設定項目と概要

項目	説明
run	0〜2 の値を設定する。0 は KSM を停止。1 は KSM を起動。2 は KSM を停止し、マージしたページは複製を作成したうえでページ共有を解消する。
sleep_millisecs	KSM がスキャンする際のインターバルをミリ秒単位で指定する。
pages_volatile	KSM が管理するツリーに登録されたページ数になる。KSM は MADV_MERGEBLE で指定したページを専用の管理ツリーに登録する。このページ数はスキャン対象のページ数でありマージが可能であるかは確認されていない。スキャンされマージの確認がされるとこの値は減る。
pages_unshared	この値は現在マージされていないページ数を示す。 MADV_MERGEBLE 指定されたページの中で同一内容のページが無いためマージできていないページである。
pages_to_scan	一度にスキャンするページ数を指定する。MADV_MERGEBLE で指定されたページのスキャンとマージ可能か確認する際のスキャンに反映される。
pages_sharing	ページ共有の利用数を示す。 例:100 ページが 1 ページにマージされていれば 100 と示す。
pages_shared	共有状態となっているページ数。 例:100 ページが 1 ページにマージされていれば値は 1 を示す。
max_kernel_pages	検査できる最大のページ数。
full_scans	スキャンした回数を示す。

一タイプの仮想機械が存在している状況があげられる [2]。

4. 検証方法

我々は 2 章と 3 章で述べた技術を踏まえ Linux KVM による仮想化環境において稼働する仮想機械のメモリ領域が KSM によってマージされることによる有効性と性能を検証することにした。3 章で述べた KSM の動作を踏まえ仮想機械の仮想メモリと実メモリの関係より、実メモリ部分は KVM によって管理される。そこで、KSM が機能する。KSM がマージする匿名ページ領域はスワップアウトしないためメモリ上に存在しつづけることになる。そのため、マージした結果圧縮効果が期待できる。従ってユーザープロセス空間内 (KVM プロセス) のコード部分とデータ部分以外即ち匿名領域に確保されるスペースが、通常はどの程度であり、KSM が有効である場合平均的にどのように縮小されていくかをホスト側のメモリを観察することによって傾向が観測できると考えられる。そこで、本論文は以下の項目についてホスト側のメモリ量を概観してみることにする。

4.1 焦点 1

仮想機械上で OS だけが動作している状況において、単一と複数の場合で KSM が無効の状態と有効の状態でのメモリ量を概観する。複数の仮想機械を稼働させる上では単一の時と同じものを用いることで、KSM によってマージされる傾向を観測する。

4.2 焦点 2

仮想機械上の OS 内で特定のアプリケーションを動かすことによって単一の仮想機械、複数の場合での KSM を無効にしている状態と有効にしている状態を観測する。仮想機械上の OS 上でアプリケーションが動作している状況を想定するためである。焦点 1 と同様に複数の仮想機械を動作させるときは同一の仮想機械を複製して用いる。焦点 2 を観測することによって KSM の性能がアプリケーションに依存するかどうかを観測できると考えられる。

4.3 焦点 3

仮想機械上の OS 内で意図的に匿名ページ領域を使うアプリケーションを動かしてみる。焦点 1,2 と同様に単一の場合と複数の場合で観測を行う。観測結果を焦点 1、2 と比較することで KSM がマージする傾向と特徴を観測するためである。また、匿名ページ領域を使うアプリケーションは匿名ページ領域をランダムに書き込むパターンと同一内容で書き込むパターンの 2 種類を用意して行うこととした。

5. 検証環境

検証環境の KVM ホスト側仕様を表 2 に示す。

表 2　ホスト側検証環境

CPU	Intel Core2 Quad CPU Q8400 2.66GHz
メモリ	8GB
OS	Debian GNU/Linux 6.0
カーネル	2.6.32-5-amd64
qemu-kvm	0.14.0
KSM	カーネル 2.6.32-5 時点のもの

ゲスト側の表 3 の仕様に従って 4 台構築した。

表 3　ゲスト側検証環境

CPU	QEMU Virtual CPU version 0.12.5 2.66GHz
メモリ	1GB
OS	Debian GNU/Linux 6.0
カーネル	2.6.32-5-686

KSM の設定はデフォルトのままとした。表 4 に示す。

表 4　KSM 設定項目

pages_to_scan	100
sleep_millisecs	20

以上の検証環境下において、仮想機械が 1 台から 4 台の

Vol.2012-DPS-152 No.32
Vol.2012-GN-85 No.32
Vol.2012-EIP-57 No.32
2012/9/14

場合に置いてのメモリ総量を焦点 1 から 3 に対応した形で観測を行うこととした。観測の際には KSM によってメモリが縮む様子が収束するまで待つことに留意した。メモリ総量はホスト側にて集計することとした。

6. 検証結果

焦点 1 の結果を図 2 に示す。4 章で述べた通り焦点 1 では仮想機械を起動させただけの状態においてホスト側にて収集したメモリ量である。単位は MB である。図中グラフ黒色が KSM を無効にしている状態、灰色が KSM を有効にした状態においてのメモリ量である。グラフは左から仮想機械が 1 台から 4 台稼働している状態である。

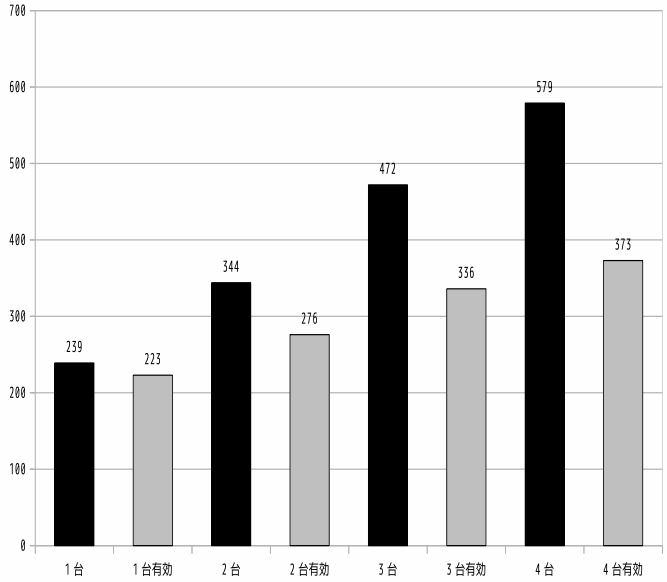

図 2 焦点 1 による観測結果

焦点 1 から観測できたことは、仮想機械が 1 台 2 台と増えていくにつれマージ率が上がっている点である。これは稼働している仮想機械が同一のものであるため匿名ページに該当する領域が多くなるためマージ対象が増加したと見受けられる。また、焦点 1 の検証の際にマージ前とマージ後の該当仮想機械プロセスのページフォールトの監視を perf によって行った。perf は Linux のパフォーマンス解析ツールであり、カーネル内のイベントや、ハードウェアイベントの発生回数を観測できる。ページフォールト回数はマージ前マージ後ともに変わらなかった。ページフォールトその物はマイナーページフォールトのみが発生していた。KSM を有効にした結果、仮想機械の性能が落ちていないと考えられる。

6.1 焦点 2 に対する検証結果

焦点 2 に対する検証結果を図 3 に示す。焦点 2 の検証においては検証用に素数を探索するアプリケーションを作成しそれぞれ仮想機械が 1 台から 4 台稼働している状況下において KSM によってメモリ総量がどの程度マージするかを観測した。図 3 に観測結果を示す。図中、黒色はマージ前のメモリ総量。灰色はマージ後のメモリ総量である。

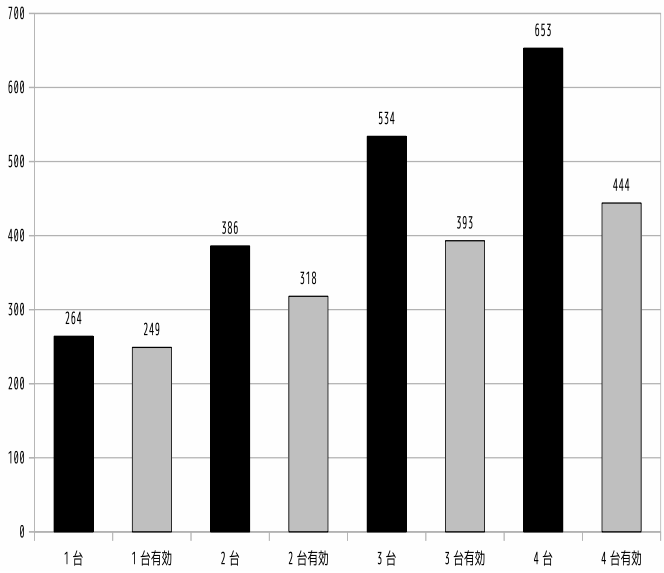

図 3 焦点 2 による観測結果

素数探索プログラムを動作させたため若干のメモリ量の増加は見られるが焦点 1 に近い結果が得られた。これは素数探索プログラムが CPU に負荷がかかる構造のプログラムになっていたため、KSM によるマージ結果に影響を与えなかったと考えられる。KSM が匿名ページのみを対象にしているためである。焦点 2 の検証に際しても perf を利用して対象仮想機械のページフォールトの監視を行った。ページフォールト回数はマージ前、マージ後ともに変わりがなかった。また、ページフォールトはマイナーページフォールトのみ発生していた。この結果から KSM を有効にした結果仮想機械の性能が劣化したとは見られない。

6.2 焦点 3 に対する検証結果

焦点 3 に対する検証結果を述べる。焦点 3 では仮想機械内の匿名メモリを意図的に利用するアプリケーションを作成した。それぞれ仮想機械内の匿名メモリ領域を乱数によってランダムなデータを書き込んだ物と匿名メモリ領域を同一のデータを書き込む 2 種類のアプリケーションである。焦点 1,2 と同様に仮想機械を 1 台から 4 台稼働させている状況での図 4 に仮想機械内の匿名メモリを乱数によってランダムなデータを書き込んだ場合においての結果を示す。図中の黒色は焦点 1、2 と同様に黒色がマージ前のメモリ総量。灰色がマージ後のメモリ総量となっている。

Vol.2012-DPS-152 No.32
Vol.2012-GN-85 No.32
Vol.2012-EIP-57 No.32
2012/9/14

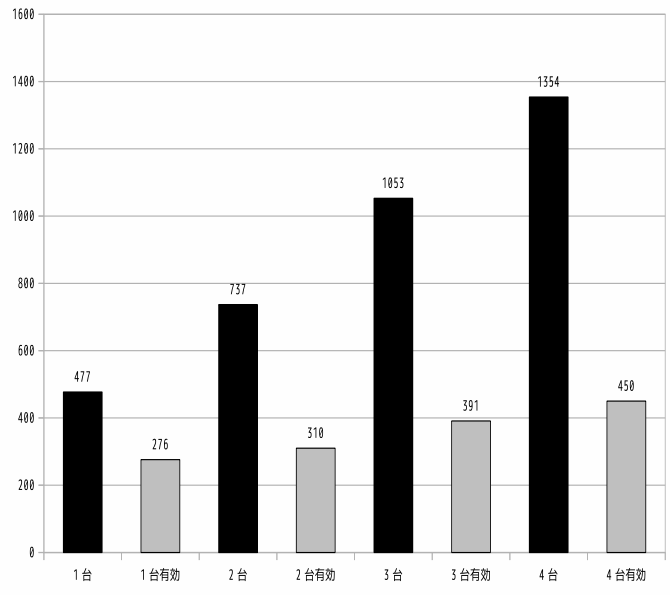

図 4 焦点 3、乱数で匿名メモリを汚した場合による観測結果

マージ前のメモリそう量が焦点 1、2 に比べて増加しているのは意図的に匿名メモリを利用するアプリケーションを利用するアプリケーションを動かしたためである。マージ後のメモリ総量に注目すると焦点 1,2 に近い数値までマージされている。これは、乱数でランダムに書き込んだ領域以外の匿名メモリ領域はマージされたと考えられる。次に仮想機械内の匿名メモリに同一データを書き込んだ場合の観測結果を図 5 に示す。

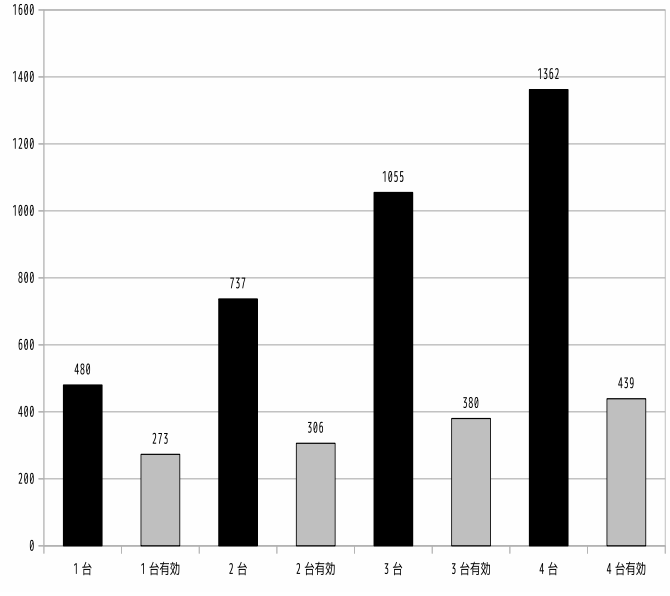

図 5 焦点 3、匿名メモリに同一データを書き込んだ場合の観測結果

こちらも図 4 の結果と同様に仮想機械内の匿名メモリ領域を意図的に確保しているため焦点 1,2 に比べメモリ量が増加している。しかし、マージ後の結果を見ると焦点 2 の結果に近い値までメモリが圧縮されていることがわかる。

さらに図 4 の結果と比較した場合図 5 のほうが 10MB 弱多くマージされている。これは後者の方で確保した匿名メモリに同一データを書き込んだため乱数を書き込んだ分だけ差が出たと考えられる。従ってアプリケーションの種類、データ構造によって意図的に匿名メモリを利用する形をとった方が KSM の恩恵を受けられ更にメモリ資源を有効に使えられると考えられる。焦点 3 においても perf を利用してマージ前とマージ後のページフォールト回数を監視した。焦点 1、2 と同様にページフォールト回数に変動は無くすべてすべてマイナーページフォールトのみが発生していた。これより KSM によるシステムの性能劣化は起こっていないと考えられる。メモリ資源に十分に余裕のある状況で検証を行っていたためにこの様なページフォールト回数になったと考えられる。従って、スワップアウトをしていないことを踏まえるとすべての仮想機械の領域がメモリ上に存在していると考えられる。KSM が匿名ページをマージしたことによってメモリ圧縮の効果が得られたと見ることができる。

7. おわりに

KSM を利用した結果ホスト側にてメモリ使用量が低下することが分かった。メモリ使用量が低下することによって同一コンピュータ内に多くの仮想機械を稼働させられると考えられる。仮想機械上で稼働する OS のメモリの使用の仕方 (アプリケーションの特性) によってもマージ率が変わってくることが分かった。本検証はホスト側のメモリ資源に余裕のある状況下だった。そのため今後検証は仮想機械の数を増やし、マージする傾向を更に観測する。これによって仮想機械の振る舞い別にメモリが低下する傾向を捉えられれば同一コンピュータ内にどれだけの仮想機械を稼働させることができるかをモデル化できると考える。

参考文献

[1] Andrea Arcangeli. Izik Eidus. and Chris Wright.: *Increasing memory density by using KSM*, Proceedings of the Linux Symposium (2009).
[2] linux-kvm.com：Using KSM (Kernel Samepage Merging) with KVM — KVM - The Linux Kernel-Based Virtual Machine，入手先 ⟨http://www.linux-kvm.com/content/using-ksm-kernel-samepage-merging-kvm⟩
[3] 土居範久　監訳：オペレーティングシステムの概念，共立出版 (2010).
[4] wikipedia：Redblack tree，入手先 ⟨http://en.wikipedia.org/wiki/Red_Black_tree⟩