# Avaliação da Efetividade dos Mecanismos de Compartilhamento de Memória em Hipervisores

Fellipe Medeiros Veiga
Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba PR
fellipeveiga@gmail.com

Carlos A. Maziero
Universidade Federal do Paraná (UFPR)
Curitiba PR
maziero@inf.ufpr.br

***Resumo*—Ambientes de virtualização de larga escala, como os usados em *datacenters* de nuvens computacionais, necessitam de um gerenciamento eficiente dos recursos computacionais utilizados. Um dos recursos mais exigidos nesses ambientes é a memória RAM, que costuma ser o principal fator limitante em relação ao número de máquinas virtuais que podem executar sobre o mesmo *host* físico. Recentemente, hipervisores trouxeram mecanismos de compartilhamento transparente de memória RAM entre máquinas virtuais, visando diminuir a demanda total de memória no sistema. Esses mecanismos "fundem" páginas idênticas encontradas nas várias máquinas virtuais em um mesmo quadro de memória física, usando uma abordagem *copy-on-write*, de forma transparente para os sistemas convidados. O objetivo deste estudo é apresentar uma visão geral desses mecanismos e também avaliar seu desempenho e efetividade. São apresentados resultados de experimentos realizados com dois hipervisores populares (VMWare e KVM), usando sistemas operacionais convidados distintos (Linux e Windows) e cargas de trabalho diversas (sintéticas e reais). Os resultados obtidos evidenciam diferenças significativas de desempenho entre os hipervisores, em função dos sistemas convidados, das cargas de trabalho e do tempo.**

***Keywords*—Gerência de memória, Compartilhamento de memória, virtualização, nuvens computacionais.**

## I. Introdução

A virtualização de sistemas consiste em executar simultaneamente vários sistemas operacionais e suas respectivas aplicações sobre um mesmo equipamento físico. Cada máquina virtual (ou sistema convidado – *guest system*) funciona como um sistema autônomo e independente do sistema físico subjacente (*host system*). A virtualização é construída por uma camada de software denominada hipervisor, responsável por mapear os recursos do sistema físico em recursos virtuais usados pelos sistemas convidados, além de assegurar o isolamento das máquinas virtuais entre si e em relação ao sistema físico.

Um dos recursos mais exigidos em ambientes virtualizados é a memória RAM, que costuma ser o principal fator limitante em relação ao número de máquinas virtuais que podem executar sobre um mesmo *host* físico. Dentro de cada máquina virtual, o sistema operacional convidado gerencia sua própria memória RAM, usando diversas técnicas. Entretanto, esse gerenciamento se limita ao escopo da própria máquina virtual, que é isolada das demais pelo hipervisor.

Partindo da hipótese de que máquinas virtuais executando sistemas operacionais convidados e/ou aplicações similares podem ter muitas páginas de memória idênticas entre si, alguns hipervisores implementaram mecanismos de compartilhamento transparente de memória RAM entre suas máquinas virtuais, visando diminuir a demanda total de memória RAM no sistema físico [1]. Ao reduzir a demanda de memória física por cada máquina virtual, mais sistemas podem potencialmente ser alocados em um dado servidor, o que resulta em uma melhor utilização do *hardware* disponível [2].

Os objetivos deste estudo são: a) apresentar uma visão geral dos mecanismos de compartilhamento de memória RAM entre máquinas virtuais oferecidos por hipervisores de mercado, b) avaliar o desempenho desses mecanismos, medindo as taxas de compartilhamento obtidas e o custo computacional exigido, e c) avaliar a efetividade dos mesmos, comparando o potencial teórico de compartilhamento com o compartilhamento real obtido por cada um. São apresentados resultados de experimentos realizados com dois hipervisores populares (VMWare e KVM), usando sistemas operacionais convidados distintos (Linux e Windows) e cargas de trabalho diversas (sintéticas e reais). Os resultados obtidos evidenciam diferenças significativas de desempenho entre os hipervisores, em função dos sistemas convidados, das cargas de trabalho e do tempo de execução.

Este artigo está estruturado da seguinte forma: a Seção II apresenta os mecanismos de compartilhamento de memória RAM implementados em hipervisores de mercado; a Seção III enumera os principais trabalhos sobre o compartilhamento de memória entre máquinas virtuais; a Seção IV apresenta a metodologia de experimentação empregada neste estudo; a Seção V compara o nível de compartilhamento obtido em cada hipervisor com o potencial teórico de compartilhamento obtido através da análise das imagens de memória das máquinas virtuais; a Seção VI analisa o comportamento dinâmico dos mecanismos de compartilhamento dos hipervisores estudados; finalmente, a Seção VII traz as conclusões do estudo.

## II. Compartilhamento de Memória em Hipervisores

Uma das principais tarefas de um sistema operacional é gerenciar a memória RAM disponível, assegurando que cada processo tenha a quantidade de memória necessária. Como a memória é um recurso relativamente escasso (em relação aos demais recursos), muitas técnicas foram desenvolvidas para otimizar seu uso, como a memória virtual, bibliotecas dinâmicas compartilhadas, carga de páginas sob demanda, alocação com *copy-on-write*, alocação preguiçosa e compressão de memória [3]. Várias dessas técnicas visam evitar páginas físicas redundantes, ou seja, que tenham o mesmo conteúdo.

Nesses casos, as páginas virtuais que têm um mesmo conteúdo são mapeadas em uma única página física (*frame*), reduzindo a quantidade de memória real usada pelo sistema.

Em um sistema virtualizado, cada máquina virtual recebe do hipervisor uma fração da memória da máquina, gerenciada pelo sistema operacional convidado. Os mecanismos internos do SO convidado se limitam à memória da própria máquina virtual, que é isolada das demais pelo hipervisor. Contudo, é de se imaginar que máquinas virtuais executando os mesmos sistemas operacionais, bibliotecas e/ou aplicações (situação típica em ambientes de *clouds*) possuam muitas páginas idênticas entre si, que poderiam ser compartilhadas.

Partindo dessa hipótese, alguns hipervisores implementaram mecanismos de compartilhamento de memória entre as máquinas virtuais, visando diminuir a demanda total de memória no sistema físico [1]. Os mecanismos de compartilhamento implementados nos hipervisores "fundem" páginas idênticas das várias máquinas virtuais em um único quadro de memória da máquina, usando uma abordagem *copy-on-write*, de forma transparente para os sistemas convidados. Essa abordagem permite eliminar páginas de memória redundantes que não seriam detectadas pelos sistemas operacionais convidados, uma vez que ocorrem entre máquinas virtuais distintas [4].

Alguns termos usuais devem ser claramente definidos para uma melhor compreensão dos mecanismos: *memória virtual da VM* diz respeito à memória vista pelos processos da máquina virtual (páginas); *memória física da VM* é a memória oferecida pelo hipervisor e gerenciada pelo sistema operacional da máquina virtual; memória de máquina (ou *host memory*) é a memória RAM real disponível no *hardware* e gerenciada pelo hipervisor [5]. O compartilhamento de páginas pode ser *intra-VM*, quando ocorre entre páginas idênticas dentro de uma mesma máquina virtual, ou *inter-VM*, quando ocorre entre páginas idênticas em máquinas virtuais distintas. Deve-se observar que, enquanto um sistema operacional convidado só consegue gerenciar o compartilhamento intra-VM, o hipervisor pode tratar ambos [2].

Hipervisores populares de código aberto ou comerciais têm mecanismos de compartilhamento baseados na estratégia *Content-Based Page Sharing* (CBPS), na qual a memória é varrida periodicamente em busca de páginas replicadas, que são mapeadas pelo hipervisor em uma única página na memória de máquina, de forma transparente aos sistemas operacionais convidados (Figura 1). Caso uma máquina virtual tente escrever em uma página assim compartilhada, o mecanismo *copy-on-write* entra em ação, desfazendo o compartilhamento para aquela máquina virtual.

O hipervisor *VMWare ESX Server* implementa o mecanismo *Transparent Page Sharing* (TPS), inicialmente proposto no sistema *Disco* [7]. Esse mecanismo é ativado quando o sistema tiver capacidade de processamento ociosa, devido ao seu elevado custo computacional [5]. Outro aspecto importante do TPS é que ele somente verifica páginas com 4 KB de tamanho, ignorando super-páginas (*huge pages*, com 2 MB), pois o custo de comparação das super-páginas é elevado e a probabilidade de encontrar réplicas diminui consideravelmente.

Para tornar mais eficiente a busca de páginas idênticas pelo hipervisor, para cada página de memória física das VMs é mantido um *hash* atualizado de seu conteúdo. Páginas com o mesmo *hash* são comparadas byte a byte; caso sejam idênticas, podem ser mapeadas em uma única página da memória de máquina, liberando as demais réplicas [8]. Como as operações de varredura, cálculo de *hash* e comparação de páginas demandam muito processamento, o intervalo de varredura é definido de forma a não degradar o desempenho do sistema [5].

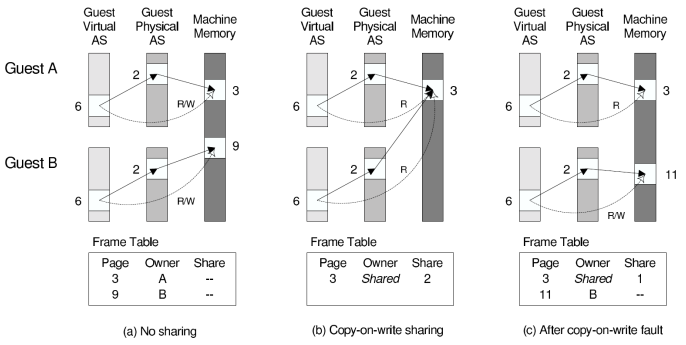


Figura 1. Content-Based Page Sharing [6].

O núcleo Linux abriga um hipervisor nativo chamado KVM (*Kernel-based Virtual Machine*) [9], que implementa um mecanismo de CBPS denominado *Kernel Samepage Merging* (KSM) [10]. Embora o KSM tenha sido desenvolvido no contexto de máquinas virtuais, ele também pode ser usado em sistemas convencionais (não virtualizados), para o agrupamento (*merging*) de páginas de aplicações. O KSM usa informações providas através da chamada de sistema `madvise` (*memory advise*), que permite a uma aplicação (ou máquina virtual) indicar ao hipervisor que uma determinada área de sua memória é apropriada ao compartilhamento.

A dinâmica do KSM se assemelha ao VMWare: um *daemon* periodicamente varre a memória buscando páginas duplicadas, que são agrupadas usando a estratégia *copy-on-write*. Entretanto, a comparação entre páginas ocorre de forma distinta: ao invés de calcular e comparar *hashes* de páginas, o KSM mantém duas árvores de páginas: a árvore *instável*, que contém páginas candidatas a serem agrupadas, mas cuja estabilidade de conteúdo ainda está sendo avaliada (através de *checksums*), e a árvore *estável*, que contém páginas já agrupadas. Uma página candidata é inicialmente comparada com as demais páginas na árvore estável: caso uma réplica seja encontrada, o agrupamento é feito; caso contrário, a página candidata é comparada às paginas na árvore instável. Caso uma réplica seja encontrada e seus *checksums* não tenham mudado, as páginas são agrupadas e movidas para um novo nó na árvore estável; caso contrário, são consideradas más candidatas ao agrupamento e removidas da árvore instável.

## III. TRABALHOS RELACIONADOS

Esta seção apresenta alguns trabalhos de pesquisa visando avaliar a eficiência dos mecanismos de compartilhamento de páginas implementados em hipervisores. O estudo [2] avaliou o potencial teórico de compartilhamento de memória em um ambiente virtualizado, através da análise de imagens de memória RAM de um conjunto de máquinas virtuais. Foram calculados os potenciais de compartilhamento de memória *intra-VM* (dentro de cada máquina virtual) e *inter-VM* (entre máquinas virtuais distintas). Em teoria, o compartilhamento

*intra-VM* pode ser aproveitado pelos sistemas operacionais convidados, mas o compartilhamento *inter-VM* só pode ser explorado pelo hipervisor. Entretanto, esse trabalho não traz nenhuma informação sobre a utilização efetiva desses potenciais de compartilhamento por hipervisores reais.

O trabalho [11] avalia o desempenho do hipervisor KVM com cargas de trabalho reais e *benchmarks* SPEC; os dados de compartilhamento foram extraídos das máquinas virtuais por um módulo de núcleo dedicado. Esse trabalho conclui que muitas possibilidades de compartilhamento são ignoradas pelo KVM, provavelmente devido à sua curta duração. O estudo [9] também avalia o desempenho do KVM, estimando o impacto das configurações do mecanismo KSM sobre o uso de memória e processador, através de cargas homogêneas e heterogêneas nas máquinas virtuais.

Nosso estudo difere de [9], pois esse trabalho apresenta cenários para recuperação de uma quantidade de memória suficiente para evitar o uso excessivo de memória e não o compartilhamento máximo possível, nesse estudo é avaliado somente o hipervisor KVM e não é feito o comparativo com outros hipervisores, o que também é verificado em [11]. Com relação ao estudo [2], este estimou apenas o potencial teórico de compartilhamento por meio de imagens de memória das máquinas virtuais, sem avaliar o compartilhamento efetivo alcançado por hipervisores reais.

## IV. METODOLOGIA

Este trabalho visa avaliar a efetividade dos mecanismos de compartilhamento transparente de memória em hipervisores de mercado. Para tanto, dois estudos serão efetuados: em um primeiro momento (Seção V), será avaliado o nível real de compartilhamento de memória alcançado por cada hipervisor em um ambiente com várias máquinas virtuais. Esse resultado será comparado com o potencial teórico de compartilhamento em cada situação, obtido usando as técnicas baseadas em *hash* descritas em [2]. O segundo estudo, apresentado na Seção VI, consiste em observar o comportamento temporal dos níveis de compartilhamento de memória nos hipervisores em várias situações de carga distintas, juntamente com o processamento demandado pelos mesmos ao longo do tempo, em cada situação.

As próximas subseções descrevem com mais detalhes os objetos de estudo, o ambiente de experimentação, os cenários experimentais e as cargas de trabalho consideradas.

### A. Hardware, hipervisores e máquinas virtuais

Todos os experimentos foram realizados em um servidor Dell R620 com as seguintes características: processador Intel Xeon E5-2609 com 8 núcleos, 2,5 GHz, cache 2,5 MB por núcleo; 16 GB de memória DDR3 (1,33 GHz).

Foram considerados hipervisores nativos (*bare metal*) com suporte para sistemas operacionais convidados Debian GNU/Linux e Microsoft Windows, tipicamente usados em *datacenters* de nuvens computacionais e na consolidação de servidores. Obviamente, os hipervisores escolhidos devem oferecer mecanismos de compartilhamento transparente de memória. Em consequência, foram escolhidos para este estudo os hipervisores VMware ESXi (versão 5.5.0 build-2068190) e KVM (versão 1:1.1.2). O hipervisor *Xen* não foi considerado, pois sua implementação de CBPS (chamada *Difference Engine*) é experimental e não está presente nas versões do hipervisor disponíveis ao público [12]. Finalmente, o hipervisor *Hyper-V* da Microsoft não implementa mecanismos de compartilhamento transparente de memória.

Para as máquinas virtuais, foram escolhidos três sistemas operacionais distintos: Windows 2012, Debian GNU/Linux 7.6 e CentOS 7.0, todas em 64 bits e configuração servidor. Dois deles (Debian e CentOS) são propositalmente similares, para avaliar o impacto dessa similaridade de núcleo, bibliotecas e ferramentas nos níveis de compartilhamento. As máquinas virtuais foram instanciadas inicialmente no hipervisor VMWare, sendo em seguida copiadas para o KVM com as mesmas configurações, para garantir igualdade de condições nos experimentos.

### B. Cenários Experimentais

Os testes foram organizados em dois cenários distintos, com as seguintes características:

I) Todas as máquinas virtuais executam o mesmo sistema operacional, neste caso o Debian GNU/Linux; aqui espera-se obter um alto nível de compartilhamento, devido à forte similaridade entre as VMs;
II) idem, mas executando o sistema operacional Windows Server 2012;

Tendo em vista a quantidade de memória de máquina disponível (16 GB) e a memória necessária para executar cada máquina virtual sem ocorrência de paginação (*swapping*), decidiu-se lançar 8 máquinas virtuais em cada experimento, com até 2 GB de memória RAM para cada uma. Os sistemas usam somente páginas de 4 KB em todos os experimentos (super-páginas foram desabilitadas).

### C. Cargas de Trabalho

Para os experimentos foram considerados dois tipos de carga de trabalho, ou seja, de aplicações executadas dentro das máquinas virtuais: uma *carga sintética*, basicamente uma aplicação que aloca e acessa continuamente uma grande área de memória, e uma *carga real*, consistindo em uma aplicação real típica de ambiente de nuvem computacional.

A carga sintética consiste em alocar uma grande área de memória (1 GB), preenchê-la com valores predefinidos e reescrever esses mesmos valores periodicamente. A área alocada deve ser alinhada com um início de página (4.096 bytes), para que as páginas da área alocada tenham o mesmo conteúdo em todas as máquinas virtuais. Essa alocação alinhada é obtida através da chamada `memalign` no Linux e `_aligned_malloc` no Windows. A carga sintética usada nos experimentos segue o algoritmo 1.

Observa-se que todas as páginas da área alocada são idênticas, e que seu conteúdo se mantém constante, pois as escritas nessa área escrevem sempre os mesmos valores nas posições de memória (linhas 4 e 11 do Algoritmo 1). Contudo, a frequência de escritas é variável, sendo definida pelo parâmetro *write_rate* ($wr$). Assim, a carga sintética pode ser configurada para diferentes níveis de *stress* no uso da memória RAM, variando o valor de $wr$. Os seguintes casos são considerados:

**Algorithm 1** Carga de trabalho sintética.

**Require:** *page_size*: size of each page (4096 bytes)
**Require:** *write_rate*: % of pages to write in each cycle
1: *buffer_size* = 1 GB
2: *buffer* = `memalign` (*page_size*, *buffer_size*)
3: **for** $i = 1$ **to** *buffer_size* **do**
4: *buffer*[$i$] = $i$ mod 256
5: **end for**
6: *num_pages* = *buffer_size* ÷ *page_size*
7: *num_writes* = *num_pages* × *write_rate*
8: **loop**
9: **for** $i = 1$ **to** *num_writes* **do**
10: *pos* = random (1...*buffer_size*)
11: *buffer*[*pos*] = *pos* mod 256
12: **end for**
13: sleep (10s)
14: **end loop**

- **Melhor caso**: considera $wr = 0$, ou seja, após o preenchimento inicial (linhas 3-5), a área de memória alocada não é mais acessada, pois $wr = 0 \Rightarrow$ *num_writes* $= 0$ (linha 7). Neste caso, o mecanismo de compartilhamento está totalmente livre para vasculhar a memória e agrupar páginas replicadas, pois as máquinas virtuais estarão ociosas.
- **Pior caso**: com $wr = 1$, cada ciclo de escrita (linhas 9-12) reescreve *num_pages* posições aleatórias de memória na área alocada (em média uma escrita por página). As contínuas escritas na memória, embora não alterem seu conteúdo, podem inibir a atividade do mecanismo de compartilhamento.
- **Casos intermediários**: com $0 < wr < 1$, cada ciclo de escrita reescreve $wr \times$ *num_pages* posições aleatórias de memória na área alocada. Após alguns testes, os valores $wr = 1/8$ e $wr = 1/32$ foram escolhidos para este estudo.

Como carga real para os experimentos, foi escolhido um servidor de banco de dados *MySQL* (versão 5.5) e um programa de *benchmark* para o mesmo denominado *MySQLslap* (versão 5.5) [13]. Essa escolha se justifica por ser uma aplicação frequentemente virtualizada em ambientes de larga escala, por apresentar uma demanda significativa de memória RAM e por estar disponível nos três sistemas operacionais utilizados.

O programa *MySQLslap* cria uma base de dados onde efetua operações de criação de tabelas, consultas, inserção e remoção de dados, para testar o comportamento e o desempenho do serviço MySQL. Após alguns testes, foram adotados os seguintes parâmetros de *benchmark*: nível de concorrência: 400 clientes simulados; iterações: 50 testes; número de consultas: 5500 por cliente; motor de SQL: *InnoDB*. A definição desse parâmetros buscou obter um grande consumo de memória e processador, evitando a ocorrência de *swapping*. Nos testes realizados, cada máquina virtual executa uma instância do cliente *MySQLslap* e do servidor *MySQL*, para minimizar o tráfego de rede e evitar interferências nos experimentos.

Durante a realização dos testes, os valores referentes ao consumo de memória, CPU e compartilhamento de memória foram coletados para posterior análise. As medições foram feitas em intervalos de 5 minutos, durante 1 hora, sendo que a primeira coleta foi efetuada imediatamente antes de lançar a respectiva carga de trabalho. Todos os dados foram coletados a partir do sistema hospedeiro, usando ferramentas de monitoração específicas de cada hipervisor: `esxtop` no VMWare, `vmstat`, `sar` e `/sys/kernel/mm/ksm` no KVM. Todos os experimentos foram realizados com as máquinas virtuais devidamente inicializadas, para evitar interferência dos procedimentos de inicialização (*boot*) nos dados coletados.

## V. Compartilhamento potencial e real

Buscou-se avaliar a eficiência dos mecanismos de compartilhamento de memória dos hipervisores em estudo, através da comparação entre os compartilhamentos realizados (medidos pelas ferramentas de monitoração dos hipervisores) e o potencial teórico de compartilhamento, obtido pela análise *offline* das imagens de memória das máquinas virtuais, de forma similar ao realizado em [2]. Nesta análise foram consideradas as melhores condições de trabalho dos hipervisores: cargas de trabalho sintéticas com o melhor caso ($wr = 0$, Seção IV-C), nos cenários I e II (Seção IV-B).

A fim de determinar compartilhamento potencial (ou compartilhamento máximo teórico), as máquinas virtuais foram suspensas após 20 minutos de execução e suas imagens de memória foram salvas em disco através de ferramentas específicas: `vmdumper` e `vmss2core` no VMWare, `virsh dump` e `lqs2mem` no KVM. Para simplificar a análise, para cada página de RAM das imagens obtidas foi gerado um *hash* SHA1 de seu conteúdo. A comparação de *hashes* revela páginas replicadas na memória das máquinas virtuais, permitindo calcular o conjunto mínimo de páginas necessárias para atender aquelas máquinas virtuais (número de *hashes* distintos), bem como o potencial de compartilhamento de páginas (número de *hashes* replicados). As páginas cujo conteúdo é nulo (zeros) não foram consideradas, pois não são mapeadas em memória física pelos hipervisores.

Cada uma das 8 máquinas virtuais recebe até 2GB de RAM do hipervisor, o que resulta em 524.417 páginas de 4 KB por VM. O resultado da análise de compartilhamento potencial e real é apresentado na Tabela I. As colunas "uso RAM" representam a quantidade de memória RAM necessária para as máquinas virtuais, tanto no caso teórico (considerando o máximo compartilhamento possível) quanto na prática (o valor observado em cada hipervisor). Nessa tabela, constata-se que a quantidade de memória usada na prática é bem maior que o mínimo teórico; isso ocorre provavelmente porque a igualdade entre algumas páginas pode ser de curta duração, não sendo capturada pelos hipervisores. Outra possível explicação para essa diferença seria a ineficiência do mecanismo implementado em cada hipervisor.

Outra constatação relevante é que a maior parte do potencial de compartilhamento provém de páginas replicadas na mesma máquina virtual (*intra-VM*), corroborando os resultados de [2]. Essas páginas poderiam ser mapeadas em uma única página de memória física da máquina virtual pelo próprio sistema operacional convidado, mas não o foram. Em nosso entender, isto significa que a gestão de memória RAM dos sistemas operacionais convidados não está adaptada para funcionar de forma eficiente em ambientes virtualizados; ela opera baseada

Tabela I. COMPARTILHAMENTO POTENCIAL E REAL.

| Experimento | | Potencial teórico | | | Uso de RAM real | |
|---|---|---|---|---|---|---|
| **Hipervisor** | **Cenário** | **uso de RAM** | **compart. intra-VM** | **compart. inter-VM** | **com compart.** | **sem compart.** |
| VMWare | I (8 × Debian) | 627 MB | 8.225 MB | 751 MB | 1.158 MB | 10.079 MB |
| VMWare | II (8 × Windows) | 3.296 MB | 9.003 MB | 2.202 MB | 4.708 MB | 16.384 MB |
| KVM | I (8 × Debian) | 630 MB | 8.241 MB | 757 MB | 965 MB | 10.014 MB |
| KVM | II (8 × Windows) | 3.258 MB | 8.489 MB | 2.760 MB | 4.176 MB | 16.384 MB |

no pressuposto que a quantidade de memória RAM é constante e deve ser usada em sua totalidade para melhorar o desempenho do sistema (fazendo cache do sistema de arquivos, por exemplo). Isso não é mais válido em sistemas virtualizados, pois a memória RAM usada como cache em cada máquina virtual poderia ser melhor aproveitada pelo hipervisor.

As duas últimas colunas da Tabela I apresentam o consumo de memória RAM em cada experimento, com mecanismo de compartilhamento de memória ligado ou desligado. Nos experimentos com o cenário II, esporadicamente algumas máquinas virtuais apresentaram *swapping* quando o mecanismo de compartilhamento estava desligado, indicando que a memória de máquina fora esgotada.

## VI. COMPORTAMENTO DINÂMICO

Esta seção apresenta resultados de experimentos que avaliam o comportamento de cada hipervisor ao longo do tempo, em função das cargas de trabalho e dos sistemas convidados. Eles permitem observar como os hipervisores reagem à demanda de uso de memória pelas máquinas virtuais e como os mecanismos de CBPS se comportam em função da disponibilidade de processador, uma vez que são ativados quando a carga de processamento é baixa.

As Figuras 2 e 3 apresenta a evolução temporal do uso de memória de máquina (RAM) no cenário I (8× Debian), com várias cargas de trabalho, usando respectivamente os hipervisores KVM e VMWare. Em todos os experimentos, foram feitas medições a cada 5 minutos, até 30 minutos. O primeiro ponto das curvas, em $t = 0$, corresponde ao uso de memória com as máquinas virtuais inicializadas, imediatamente antes de lançar a carga de trabalho. A linha tracejada (*Host Mem*) corresponde à memória usada pelo próprio hipervisor, que não varia de forma perceptível.

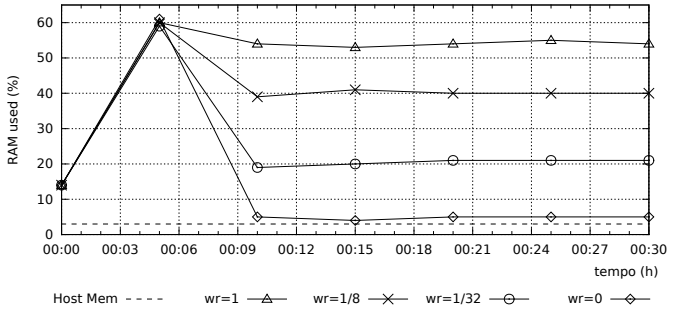


Figura 2. Uso de RAM no KVM com 8 VMs Debian.

Em ambos os gráficos percebe-se um pico no uso do memória física no início do experimento, após o lançamento das cargas de trabalho. Esse pico é mais significativo no KVM (Figura 2), cujo mecanismo de compartilhamento parece levar

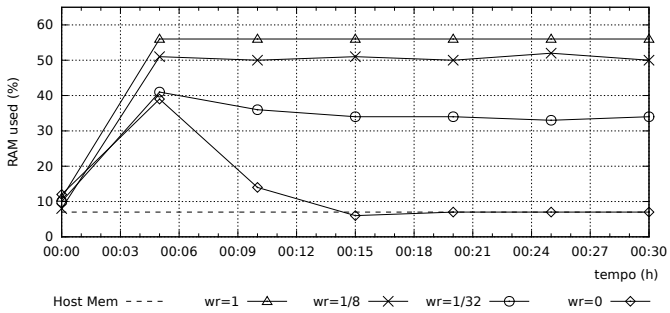


Figura 3. Uso de RAM no VMware com 8VMs Debian.

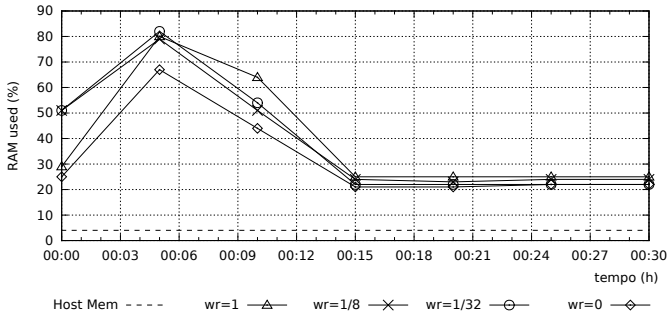


Figura 4. Uso de RAM no KVM com 8VMs Windows.

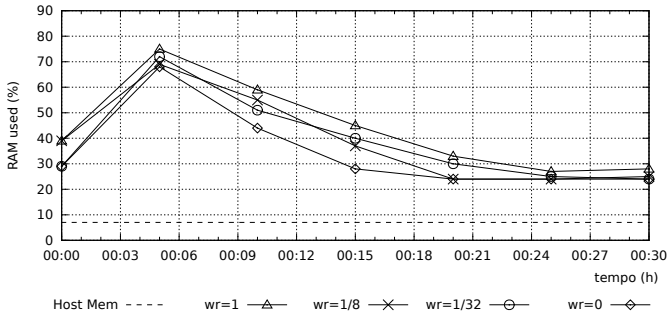


Figura 5. Uso de RAM no VMware com 8VMs Windows.

mais tempo para iniciar sua operação[1]. Já no VMWare (Figura 3) esse pico é menos intenso, sobretudo para as cargas de trabalho com menor taxa de escritas ($wr \rightarrow 0$). Contudo, após o pico inicial, o hipervisor KVM reduz de forma mais significativa o uso de memória física que o VMWare, sobretudo para as taxas de escrita mais baixas. Ambos os hipervisores têm comportamento similar no pior caso ($wr = 1$).

[1]O mecanismo KSM do hipervisor KVM somente funciona quando a quantidade de memória livre cai abaixo de um limite (por default 20%); em nosso testes, esse valor foi alterado para 75%, para poder observar melhor seu comportamento.

As figuras 2 a 5 mostram que o hipervisor KVM teve um melhor aproveitamento das possibilidades de compartilhamento de memória, sobretudo nas cargas mais leves ($wr \rightarrow 0$). Observa-se também uma clara diferença nos níveis de compartilhamento obtidos entre os cenários I ($8\times$ Debian) e II ($8\times$ Windows). Apesar do menor consumo de memória ser atingido no cenário I, este cenário também é o mais sensível à carga de trabalho ($wr$). As razões específicas para esse comportamento estão sendo investigadas.

Conforme discutido na Seção II, os mecanismos de compartilhamento de memória podem consumir muito processamento para encontrar páginas de memória replicadas. As Figuras 6 7 apresentam o uso de processador (no *host*) para os hipervisores estudados, nos cenários I e II, para as cargas de trabalho com $wr = 0$ (melhor caso) e $wr = 1$ (pior caso). Observa-se um uso maior de processador nos primeiros minutos do experimento, sobretudo para o KVM no cenário II ($8\times$ Windows), mas ele sempre permanece abaixo de 20%. Ou seja, haveria processamento disponível para uma política de compartilhamento mais agressiva.

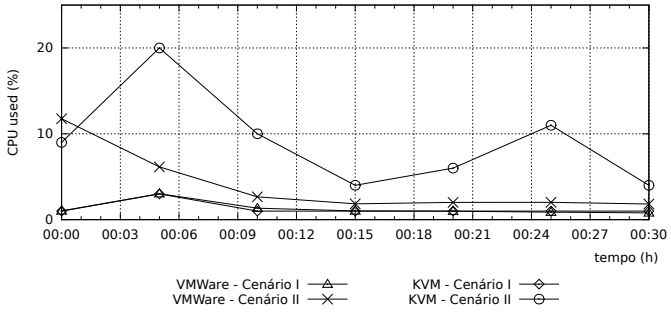


Figura 6. Uso de processador no *host* – melhor caso ($wr = 0$).

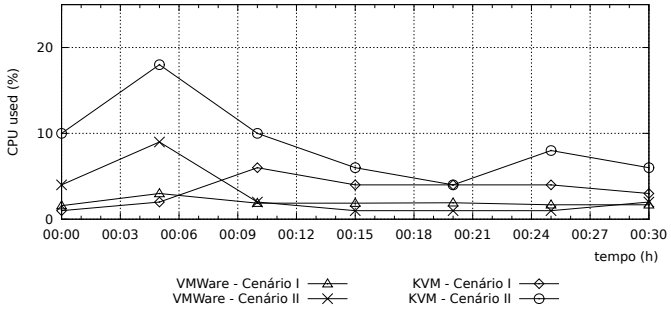


Figura 7. Uso de processador no *host* – pior caso ($wr = 1$).

## VII. Conclusão

Este estudo apresentou uma avaliação experimental do compartilhamento de memória em hipervisores comerciais executando sistemas operacionais convidados Windows e Linux. Observa-se que o compartilhamento de memória é fortemente influenciado pelas aplicações em execução e pelos sistemas operacionais utilizados nas máquinas virtuais. Nos cenários analisados o KVM apresentou melhores níveis de compartilhamento de memória; por outro lado, o VMware apresentou um uso menos intenso de CPU pelos mecanismos de busca de réplicas em páginas de memória.

Além dos experimentos apresentados neste texto, também foram estudados cenários envolvendo outros sistemas operacionais convidados e cenários mistos, com mais de um tipo de sistema operacional simultaneamente (em máquinas virtuais distintas). Os resultados obtidos nesses experimentos são similares aos aqui apresentados.

Está em andamento uma avaliação mais detalhada de ambos os hipervisores com cargas de trabalho reais (MySQL e outros serviços). Além disso, pretende-se fazer uma análise detalhada das imagens de memória de máquinas virtuais executando Linux e Windows, buscando compreender as causas das diferenças de níveis de compartilhamento observadas entre os sistemas operacionais convidados. Suspeita-se da influência de mecanismos de segurança como ASLR (*Address Space Layout Randomization*) e da quantidade de bibliotecas dinâmicas carregadas na memória em ambos os sistemas, que precisa ser investigada.

Outro trabalho futuro seria a avaliação dos hipervisores usando sistemas operacionais específicos para ambientes de nuvem computacional, como o OSv [14], que possuem um *footprint* de memória menor que os sistemas convencionais.

## Referências


[1] C.-R. Chang, J.-J. Wu, and P. Liu, "An empirical study on memory sharing of virtual machines for server consolidation," in *IEEE Intl Symposium on Parallel and Distributed Processing with Applications*, May 2011, pp. 244–249.

[2] S. Barker, T. Wood, P. Shenoy, and R. Sitaraman, "An empirical study of memory sharing in virtual machines," in *USENIX Annual Technical Conference*, Berkeley CA, USA, 2012, pp. 25–25.

[3] U. Vahalia, *UNIX Internals – The New Frontiers*. Prentice-Hall, 1996.

[4] M. Sindelar, R. K. Sitaraman, and P. Shenoy, "Sharing-aware algorithms for virtual machine colocation," in *ACM Symposium on Parallelism in Algorithms and Architectures*, 2011, pp. 367–378.

[5] "Understanding memory resource management in VMware ESX Server," VMWare Inc., Palo Alto CA, USA, Tech. Rep. EN-000411-00, 2010.

[6] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage, "Scalability, fidelity, and containment in the Potemkin virtual honeyfarm," *SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 148–162, Oct. 2005.

[7] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum, "Disco: Running commodity operating systems on scalable multiprocessors," *ACM Transactions on Computer Systems*, vol. 15, no. 4, pp. 412–447, 1997.

[8] L. Chen, Z. Wei, Z. Cui, M. Chen, H. Pan, and Y. Bao, "Classification-based memory deduplication through page access characteristics," in *10th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. New York, NY, USA: ACM, 2014, pp. 65–76.

[9] S. Rachamalla, D. Mishra, and P. Kulkarni, "Share-o-meter: An empirical analysis of KSM based memory sharing in virtualized systems," in *Intl Conference on High Performance Computing*, Dec 2013, pp. 59–68.

[10] A. Arcangeli, I. Eidus, and C. Wright, "Increasing memory density by using KSM," in *Ottawa Linux Symposium*, 2009, pp. 19–28.

[11] T. Gröninger, "Analyzing shared memory opportunities in different workloads," Master's thesis, Karlsruher Institut für Technologie, 2011.

[12] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, "Difference engine: Harnessing memory redundancy in virtual machines," *Communications of the ACM*, vol. 53, no. 10, pp. 85–93, oct 2010.

[13] "MySQLslap – Load Emulation Client," http://dev.mysql.com, accessed: 02-12-2014.

[14] A. Kivity, D. Laor, G. Costa, P. Enberg, N. Har'El, D. Marti, and V. Zolotarov, "OSv — optimizing the operating system for virtual machines," in *USENIX Annual Technical Conference*, 2014, pp. 61–72.