

Análise do Impacto da Virtualização sobre os Estados de Economia de Energia de Processadores em Ambientes de Computação em Nuvem

Guilherme Costa, Renan Alves, Daniel Cordeiro*

¹ Escola de Artes, Ciências e Humanidades
Universidade de São Paulo – São Paulo – SP – Brasil

{guilherme_costa, renanalves, daniel.cordeiro}@usp.br

Abstract. *Cloud computing leverages virtualization to optimize resource use. However, the interaction between the virtualization layer and processor power management mechanisms remains poorly understood. This paper presents a preliminary investigation into the impact of virtualization on C-states, evaluating how virtual machines influence power savings in modern processors.*

Resumo. *A computação em nuvem utiliza amplamente técnicas de virtualização para otimizar o uso de recursos computacionais. Entretanto, a interação entre a camada de virtualização e os mecanismos de gerenciamento de energia dos processadores ainda não é completamente compreendida. Este trabalho apresenta uma investigação preliminar sobre o impacto da virtualização nos C-states, avaliando como as máquinas virtuais influenciam a economia de energia em processadores modernos.*

1. Introdução

A utilização cada vez mais frequente da computação em nuvem acaba por centralizar o processamento nas grandes estruturas de *data centers*, tornando ainda mais evidente a necessidade de otimização do consumo de energia. Tendo em vista que uma parte significativa da energia consumida pelo servidor está relacionada à utilização do processador, os mecanismos de gestão de energia no nível de hardware tornam-se críticos para minimizar o consumo de energia, especialmente durante períodos de ociosidade.

Alguns desses mecanismos estão associados aos estados de economia de energia, chamados de *C-states* do processador, que são progressivamente mais ociosos. A economia de energia aumenta à medida que a profundidade do estado C aumenta, e a latência também aumenta. Entretanto, em um ambiente de virtualização, a camada de virtualização introduz uma indireção entre o convidado e o hardware, podendo interferir no funcionamento das transições entre esses estados e impedir o acesso às profundezas dos estados ociosos.

Este trabalho visa analisar a influência dos ambientes de máquinas virtuais Linux nas transições entre os *C-states* dos processadores x86 em dois cenários. O primeiro em um ambiente *bare-metal* e o segundo em um ambiente com virtualização. Dessa forma,

*Este trabalho foi realizado com apoio da FAPESP (proc. #24/19288-9), do CCD Cidades Carbono Neutro (FAPESP #24/01115-0) e do Programa Unificado de Bolsas (PUB) da USP.

a hipótese de partida deste estudo é que o sistema convidado e o hipervisor impeçam a transição para as profundidades dos *C-states*, aumentando o consumo de energia em comparação com o ambiente *bare-metal*.

2. Trabalhos Relacionados

A eficiência energética dos centros de dados está fortemente associada ao processador, que gerencia a energia nos períodos de ociosidade por meio dos *C-states*. [Sharafzadeh et al. 2019] mostram que o governador de ociosidade padrão do Linux economiza energia à custa da latência de cauda, pois estados C mais profundos demandam maior latência de saída, e propõem um governador baseado em *machine learning*. Esse estudo, porém, é realizado em *bare-metal*, sem considerar o impacto da virtualização. O caso virtualizado foi tratado por [Mvondo et al. 2021], que identificam uma lacuna semântica entre o escalonador do hipervisor e o do convidado: tratada como caixa-preta, a VM oculta do hospedeiro o que ocorre em seu interior. Em Linux/KVM com *microVMs*, as vCPUs seguem escalonadas mesmo ociosas, desperdiçando de 20% a 75% do tempo de CPU sob sobrecomprometimento de recursos. O foco é o desperdício de ciclos, não os *C-states*, mas o estudo evidencia como a virtualização distorce a percepção de ociosidade, o que motiva investigar seu impacto na economia de energia.

Vários trabalhos medem o impacto da virtualização no desempenho e no consumo de energia com múltiplas VMs. [Altahat et al. 2025] mostram que a escolha do hipervisor afeta significativamente o consumo e a vazão do servidor, e que a afinidade entre convidado e hipervisor importa. Mais próximo desta linha, [Bermejo and Juiz 2022] medem o sobrecusto de virtualização (OVv) e de consolidação (OVc), definindo o OVv como a diferença de tempo de resposta entre uma VM e o ambiente *bare-metal* — análoga ao contraste aqui estudado — e mostrando, com KVM, VirtualBox e Docker sob *sysbench* e *stress-ng*, que ele cresce com o número de camadas de software interpostas. Essas medições, porém, usam indicadores agregados (tempo de resposta, consumo total), sem isolar o mecanismo de *C-states* como variável de análise.

Percebe-se, assim, uma lacuna: a literatura de gestão da ociosidade caracteriza os *C-states* em *bare-metal* [Sharafzadeh et al. 2019]; a de virtualização evidencia a distorção da ociosidade pela lacuna semântica [Mvondo et al. 2021]; e a de sobrecusto e energia mede o impacto de forma agregada [Altahat et al. 2025, Bermejo and Juiz 2022]. Este trabalho posiciona-se na interseção desses eixos ao caracterizar empiricamente, com *sysbench* e *turbostat*, se e como a virtualização afeta o acesso aos *C-states* mais profundos em relação ao *bare-metal*.

3. Metodologia

3.1. Ambiente Experimental

Os testes foram realizados em um servidor equipado com processador Intel Xeon Gold 6230R (arquitetura Cascade Lake, 26 núcleos físicos / 52 *threads*, frequência base de 2,10 GHz), com suporte à virtualização em hardware (Intel VT-x e EPT) e 64 GB de memória. O hospedeiro operava o sistema operacional Ubuntu 24.04.4 LTS (kernel 6.8.0), utilizando a pilha KVM/QEMU gerenciada pelo libvirt [Kivity et al. 2007, Bellard 2005]. O gerenciamento de energia do hospedeiro foi mantido idêntico nos dois cenários, distinguindo-se o governador de frequência (*P-state*) `schedutil`, sob o driver

`intel_cpufreq` (isto é, o `intel_pstate` operando em modo passivo), do governador de estados ociosos (*C-state*) menu, sob o `intel_idle` [Pallipadi et al. 2007]; mantendo-se ativados o Turbo Boost e todos os *C-states*. A VM foi criada a partir de uma imagem *cloud* do Ubuntu 24.04 (*cloud-init*) com o modelo de CPU `host-passthrough`, que expõe ao convidado o mesmo modelo de processador e o mesmo conjunto de instruções da CPU física. Foram atribuídas 2 vCPUs e 2 GB de memória à VM, com as vCPUs mapeadas a núcleos físicos dedicados (`taskset/libvirt`) para evitar que migrações de vCPUs afetassem as medidas.

3.2. Carga de Trabalho

A carga é intensiva em CPU e é gerada pela ferramenta *sysbench*,¹ na opção `cpu` que calcula números primos até um determinado limite (`-cpu-max-prime`). Para assegurar a consistência nas operações, a carga é definida com base em um número de eventos (`-events`) predefinido, em vez do tempo cronológico, para que todos os cenários executem exatamente a mesma quantidade de tarefas. São alterados o número de *threads* (`-threads`) e o de repetições, levando a uma utilização que vai desde a leve (com poucas *threads*, permitindo a ociosidade dos núcleos) até a saturação (com $threads \geq$ núcleos disponíveis); a ociosidade que viabiliza as fases mais profundas dos *C-states* está diretamente ligada ao tempo de ociosidade entre os cálculos.

3.3. Coleta de Dados

A instrumentação usa o *turbostat*,² que reporta frequência, temperatura, energia (via RAPL [Khan et al. 2018]) e residência nos estados ociosos, por núcleo ou por pacote. Como esses contadores e os registradores RAPL pertencem ao processador físico e não são bem virtualizados no convidado, o *turbostat* opera sempre no *host*: os dois cenários são medidos nas CPUs físicas, pelo mesmo ponto de instrumentação. São coletados os percentuais de tempo que o processador permanece em atividade e em cada estado ocioso, tanto no nível de cada núcleo individual quanto no nível do pacote como um todo. No nível de núcleo, observam-se um estado ocioso mais raso e um mais profundo, sendo que quanto mais profundo o estado, maior a economia de energia obtida; no nível do pacote, o estado profundo equivalente só é atingido quando todos os núcleos estão simultaneamente ociosos, refletindo a coordenação necessária entre os núcleos para que a economia de energia se manifeste no processador como um todo. Além disso, é coletada a potência média consumida pelo pacote ao longo de cada execução. Ambos os cenários são medidos em várias execuções, descartando-se os *outliers* e a fase inicial de aquecimento.

3.4. Cenários Avaliados

Comparam-se dois cenários sob parâmetros de carga idênticos: (i) *bare-metal*, com a carga executada diretamente no sistema hospedeiro, e (ii) virtualizado, com a carga no convidado KVM. Ambos usam o mesmo número de núcleos ativos, os mesmos governadores no hospedeiro e os mesmos parâmetros do *sysbench*, de modo que a única diferença estrutural é a camada de virtualização. Coleta-se também uma linha de base em repouso (sem carga) para caracterizar a residência nos estados C de cada configuração em ociosidade. Se necessário, cenários adicionais variam a concorrência (número de *threads*) para avaliar se o hipervisor restringe a entrada do processador em estados mais profundos.

¹<https://github.com/akopytov/sysbench>

²<https://www.linux.org/docs/man8/turbostat.html>

3.5. Métricas de Análise

A análise baseia-se em três categorias de métricas: **desempenho** (taxa de vazão do *sysbench* em eventos por segundo, número de eventos totais e latência); **comportamento nos C-states** (tempo que o processador passa em cada estado, conforme relatado pelo *turbostat*); e **eficiência energética** (potência média do pacote e energia associada, obtida pela integração da potência ao longo do tempo, com um indicador de eficiência em eventos por joule). A comparação entre os cenários virtualizado e *bare-metal* fornece a média e o desvio padrão das execuções, buscando identificar se a virtualização reduziu a eficiência energética (ao impedir o uso dos C-states mais profundos) sem um ganho de desempenho equivalente, ou até mesmo com perda.

4. Resultados Parciais

No estágio atual, o esforço concentrou-se na construção e na validação do ambiente experimental e da infraestrutura de coleta. Foram implementados os scripts de orquestração que automatizam a criação de VMs a partir de imagens *cloud* com *cloud-init*, a execução parametrizada do *sysbench* e a captura de métricas via *turbostat* no hospedeiro. Testes de execução confirmam que a instrumentação reporta corretamente a residência em C-states e o consumo de energia, o que valida o procedimento de medição. A campanha experimental comparativa entre os cenários *bare-metal* e virtualizado está em andamento, e os resultados quantitativos serão apresentados na continuidade do trabalho.

Referências

- Altahat, M. A., Mhaidat, K., and Al-Khaleel, O. (2025). Quantitative analysis of hypervisor efficiency and energy consumption in heterogeneous multi-vm environments with varied server workloads. *Simulation Modelling Practice and Theory*, 141:103102.
- Bellard, F. (2005). QEMU, a fast and portable dynamic translator. In *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*, pages 41–46.
- Bermejo, B. and Juiz, C. (2022). A general method for evaluating the overhead when consolidating servers: performance degradation in virtual machines and containers. *The Journal of Supercomputing*, 78:11345–11372.
- Khan, K. N., Hirki, M., Niemi, T., Nurminen, J. K., and Ou, Z. (2018). RAPL in action: Experiences in using RAPL for power measurements. *ACM TOMPECS*, 3(2).
- Kivity, A., Kamay, Y., Laor, D., Lublin, U., and Liguori, A. (2007). kvm: the Linux virtual machine monitor. In *Procs. of the Linux Symposium*, volume 1, pages 225–230.
- Mvondo, D., Barbalace, A., Tchana, A., and Muller, G. (2021). Tell me when you are sleepy and what may wake you up! In *Proceedings of the ACM SoCC '21*, pages 562–569.
- Pallipadi, V., Li, S., and Belay, A. (2007). cpuidle: Do nothing, efficiently. In *Proceedings of the Linux Symposium*.
- Sharafzadeh, E., Sanaee Kohroudi, S. A., Asyabi, E., and Sharifi, M. (2019). Yawn: A CPU idle-state governor for datacenter applications. In *Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '19)*, pages 91–98.