Consumo de energia de sistemas operacionais

Shoaib Akram, Manolis Marazakis e Angelos Bilas, da Foundation for Research and Technology - Hellas (FORTH) e Institute of Computer Science (ICS), Grécia

O artigo avalia a sobrecarga relativa do sistema operacional quando são usadas máquinas virtuais em aplicações com intensas operações de entrada/saída. Os resultados indicam que o OS pode custar até 60% mais em termos de energia consumida e que uma única instância de máquina virtual custa 150% em desempenho e 180% em consumo de energia por operação.

A quantidade de dados gerados na sociedade moderna está crescendo rapidamente. Várias projeções estimam que, por volta de 2020, o mundo produzirá 35 zetabytes de dados [10]. Para gerenciar e processar esse imenso volume de informações, surgiu o conceito de aplicações centradas em dados [12]. Os data centers já consomem uma significativa quantidade de energia e com a taxa de crescimento anual de cerca de 14%, o seu consumo total de energia, nos EUA, será de 300 bilhões de kWh. Por essa razão, há uma crescente pressão para melhorar a eficiência das modernas aplicações centradas em dados.

Muitas aplicações centradas em dados usam de forma intensiva o sistema operacional (OS) para acesso à rede e armazenamento. Para muitas aplicações, a contribuição do OS para o tempo de execução total é comparável ou maior que o tempo útil do usuário, ou seja, o tempo que uma aplicação gasta na execução do código no espaço do usuário. Particularmente com relação às aplicações centradas em dados, todo o trabalho realizado pelo OS visa o fornecimento de tradução do nome, recuperação e gerenciamento de buffer e de dispositivo. Não está relacionado aos dados reais, em si, mas ao processo de mudar os dados de um armazenamento permanente para os buffers da aplicação.

Além disso, hoje as VMs - máquinas virtuais são normalmente usadas para permitir que múltiplas aplicações rodem no mesmo servidor. A melhoria do uso do servidor requer a execução de múltiplas aplicações e, para fins de isolamento, elas em geral rodam dentro de VMs separadas. Isso introduz sobrecargas adicionais para o desempenho de E/S.

Neste artigo, estamos interessados em compreender a relativa sobrecarga do OS e das VMs sobre o processamento de aplicações com intensidade de E/S. Usamos aplicações e conjuntos de dados típicos de cargas de trabalho implantadas nos atuais data centers. Rodamos as cargas de trabalho usando um OS *commodity* e medimos sua sobrecarga sozinho. Usamos um modelo simples para traduzir a divisão do tempo de execução em ciclos e a energia gasta por E/S (cpio e eio). Em seguida, rodamos as cargas de trabalho usando um VMM - monitor de máquina virtual popular e medimos a sobrecarga adicional. Também relatamos a redução do custo de energia por E/S quando se usam múltiplas VMs para rodar instâncias independentes da mesma carga de trabalho.

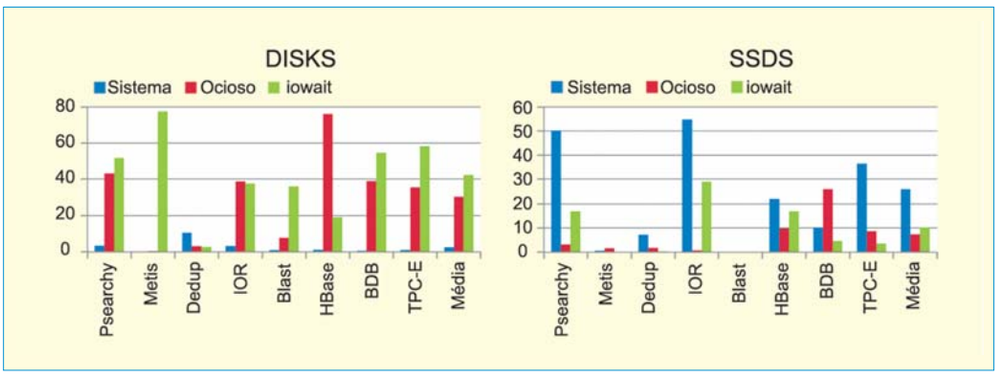

Fig. 1 - Porcentagem de tempo de execução (eixo Y) gasto como tempo de sistema, ocioso e iowait em SSDs e com discos

Metodologia

Nosso principal objetivo, neste artigo, é quantificar sobrecargas introduzidas pelas camadas de sistema. Em particular, estamos interessados na camada OS e na camada VM. As métricas do nível de aplicações, portanto, não são úteis para a análise dos componentes individuais das complexas pilhas de software de hoje em dia. Assim, propomos o uso de ciclos por operação de E/S (cpio) como métrica para quantificar as sobrecargas no nível de sistemas. Calculamos o *cpio* rodando a aplicação e medindo a divisão do tempo de execução consistindo de usuário, sistema, tempo ocioso e tempo de espera de E/S (*iowait*). A seguir, discutimos brevemente o que cada componente do tempo de execução significa.

O tempo do usuário refere-se ao tempo que uma aplicação leva para executar o código no espaço do usuário. Quando uma aplicação solicita serviços pelo OS, o tempo gasto é classificado como tempo do sistema. O tempo que uma aplicação leva esperando que uma operação de E/S pendente seja completada é chamado de tempo de espera de E/S (*iowait*). Finalmente, quando um processador não tem qualquer trabalho a realizar, o tempo é contado como tempo ocioso.

Rodamos cada carga de trabalho usando parâmetros de configuração e conjuntos de dados que resultam numa grande quantidade de E/S. Primeiro rodamos cada carga de trabalho sobre a distribuição do OS *commodity*, com o mais recente kernel Linux, e medimos a divisão do tempo de execução. Relatamos a sobrecarga do componente do sistema sozinho, em comparação com o tempo de execução total gasto para fins de processamento. Mostramos que, embora baixa em comparação com o componente do usuário, a sobrecarga do componente do sistema cresce com o número de núcleos. Finalmente, examinamos como as sobrecargas aumentam observando cpio e eio em um número crescente de núcleos.

Em comparação com a energia consumida pelo componente do usuário de uma única instância de carga de trabalho, observamos que:

- Os servidores que usam subsistemas de armazenamento baseados em disco, além dos lentos tempos de resposta, também são energeticamente ineficientes em E/S de processamento. Em particular, com as atuais pilhas do sistema, os servidores que usam subsistemas de armazenamento baseados em disco gastam até seis vezes mais energia por operação de E/S. A diferença é reduzida de seis vezes para apenas 58%, se o consumo de energia ociosa nos servidores for igual a zero.
- Em média, o tempo de execução do componente do sistema custa 60%, em termos de consumo de energia, por operação de E/S.
- A virtualização custa até 150% em termos de ciclos gastos por operação de E/S, para um conjunto de aplicações centradas em dados. Da mesma forma, com o acréscimo dos custos da camada VM há, em média, uma sobrecarga de 180%, em termos da energia consumida por operação de E/S.
- A camada OS, como é atualmente, não funciona bem com o aumento do número de núcleos. Em média, para oito cargas de trabalho centradas em dados, há um aumento de 12, 24 e 92 vezes nos ciclos do sistema por operação de E/S, em comparação com os ciclos do sistema com um núcleo.

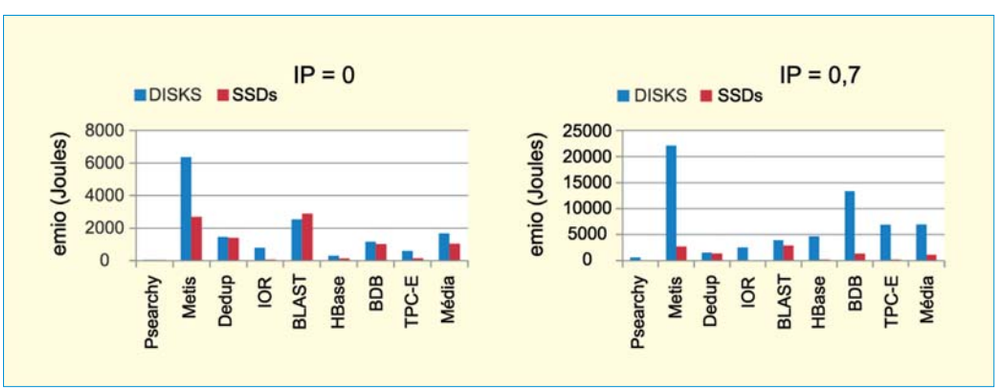

Fig. 2 - Consumo de energia por 1 milhão de E/S (emio) com discos e SSDs

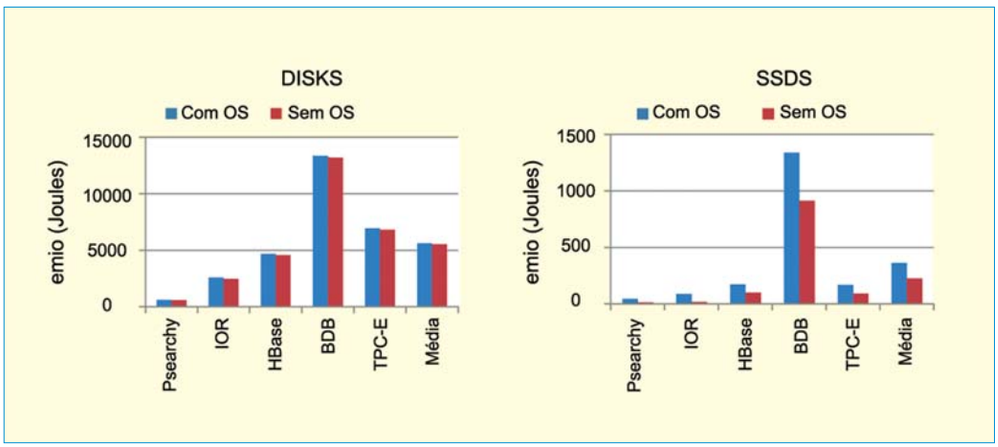

Fig. 3 - Sobrecarga da camada de sistema com discos e SSDs

Também observamos o número de operações de E/S de 512 bytes que uma aplicação realiza durante o tempo de execução. Trata-se de um tamanho típico do setor em muitos dispositivos de armazenamento. O tamanho real da operação de E/S, do ponto de vista do kernel, é diferente e, normalmente, da ordem de kilobytes. Supomos que a totalidade do volume de dados de leitura/escrita durante o tempo de execução consiste de muitos *chunks* de 512 bytes.

Em seguida, calculamos o cpio dividindo os ciclos do usuário mais os ciclos do sistema consumidos durante o tempo de execução pelo número de operações de E/S. Não rodamos as cargas de trabalho até o fim, mas o bastante para capturar o comportamento representativo. Em particular, fizemos medições durante as quais a aplicação gera o *throughput* típico se houver permissão de rodar por períodos mais longos.

Um importante benefício do cpio como métrica do nível de sistema é a facilidade de traduzir para energia por operação de E/S. Usamos um método simples para calcular a energia gasta por operação de E/S. Nosso modelo deriva-se de resultados relatados na literatura recente [9, 20, 18]. Em particular, medimos a utilização da CPU e, em seguida, o pico de energia e a potência ociosa típicos de nossas máquinas de servidor, para calcular a energia por operação de E/S. Não incluímos a energia consumida por dispositivos de armazenamento. Relatamos os resultados supondo não haver energia ociosa, o que é típico dos servidores implantados nos dispositivos de armazenamento dos data centers atuais [20].

Usamos a seguinte equação para calcular a energia por E/S (eio) em joules:

$$eio = \{P * (1 * cpio + 1P * cpio^{i})\}/(N * F)$$

onde:

- P: potência de pico do servidor (menos o subsistema de armazenamento);
- cpio: ciclos por E/S;
- IP: fração da potência de pico quando a máquina está em estado ocioso;
- N: número de núcleos; e
- F: frequência de cada núcleo.

Mostramos os resultados do consumo de energia necessário e o desempenho (e processamento) de 1 milhão de operações de E/S (emio).

Uma pilha de software pode parecer ter eficiência de cpio ou de eio, ao fazer inúmeras e pequenas operações de E/S. Por exemplo, aplicações que realizam um grande número de operações de metadados ao longo da execução podem parecer ter um baixo cpio, embora as operações E/S de metadados sejam, normalmente, 'induzidas' e não estritamente necessárias para o processamento de um conjunto de dados específico. Assim, ao observarmos o cpio e o eio, o seu mero valor absoluto não é adequado para julgar a qualidade de uma pilha ou plataforma.

Em nosso trabalho, primeiro comparamos o emio de dois servidores com diferentes subsistemas de armazenamento. Em seguida, calculamos o emio sem a camada OS. Depois, rodamos as cargas de trabalho dentro de uma VM e mostramos o aumento de emio. Em seguida, usamos a consolidação do servidor com duas VMs. Finalmente, usamos o componente de cpio do sistema para avaliar a escalabilidade das atuais pilhas OS. Medimos os ciclos do sistema por E/S para 1, 4, 8 e 16 núcleos e fazemos uma projeção usando um modelo linear de 1000 núcleos.

Aplicações

Discutimos os parâmetros de configuração, o modo de operação e os conjuntos de dados usados nas aplicações. A tabela I apresenta um resumo das cargas de trabalho usadas pelas aplicações. Em alguns casos, usamos múltiplas instâncias da mesma aplicação para formar uma carga de trabalho que utilize melhor os recursos do servidor e de armazenamento. A seguir, discutiremos brevemente cada aplicação.

O *checkpointing* é feito em aplicações de computação de alto desempenho (HPC - high performance computing) para a recuperação de possíveis falhas. Usamos IOR [16], que simula vários padrões de *checkpointing* que aparecem no domínio HPC. Devido à camada MPI, o IOR tem um tempo de usuário moderado e os E/S *in-flight* emitidos por vários processos MPI simultâneos induzem um tempo de iowait significativo, devido à contenção de recursos E/S.

A indexação de arquivos é feita, principalmente, como trabalho de *back-end* em data centers e instalações de hospedagem da web. Usamos Psearchy [15, 5] como aplicação de indexação de arquivo. Rodamos o Psearchy usando múltiplos processos, em que o processo *aach* escolhe arquivos de uma fila compartilhada de nomes de arquivos. Uma tabela *hash* é mantida por cada processo para armazenar índices BDB na memória. As tabelas *hash* são liberadas para os dispositivos de armazenamento uma vez alcançado o tamanho certo. Modificamos o Psearchy original para usar leituras em bloco em vez de leituras por caracteres para melhorar o *throughput* de E/S.

Há um aumento do tempo de sistema do Psearchy na indexação de grandes arquivos. Ao contrário, na indexação de pequenos arquivos há um aumento do tempo do usuário. Uma complexa estrutura de diretório resulta num aumento do tempo do sistema.

A deduplicação é uma técnica de compressão usada em *farms* de armazenamento e data centers. Usamos o kernel Dedup do conjunto de referência PARSEC [4]. O núcleo Dedup kernel tem cinco estágios, sendo o primeiro e o último para execução de E/S. Os três estágios intermediários usam, cada um, uma associação de *threads*. Observamos

Tab. I - Cargas de trabalho para avaliação

Aplicação	Descrição	Parâmetros	Tamanho do conjunto de dados (Gb)
IOR	*Checkpointing* de aplicação	Processos = 128; Tamanho de arquivo = 2 Gb; Modo E/S = MPLIO; *Offsetting* dentro do arquivo = sequencial	128
Psearchy	Indexação de arquivo	Hierarquia de diretório = horizontal; Tamanho de documento = 10 Mb Processos = 32; Tamanho de tabela *hash* = 128 Mb	100
Dedup	Compressão de arquivo	*Dedup:* Tamanho de arquivo = 1 Gb; Instâncias = 10; *threads* por estágio = 32	10
Metis	Biblioteca Mapreduce para máquina de núcleo único e múltiplos núcleos	Aplicação: contagem de palavras; Instâncias = 5; Tamanho de arquivo = 1 Gb; *threads* de mapa = 8; *threads* de redução = 8	20
BDB	*Key-value Store* (baseado em Java)	*Threads* = 128; Campos por registro = 10; Tamanho de campo = 1 kb	30
HBase	Armazenamento NoSQL	*Threads* = 128; Campos por registro = 10; Tamanho de campo = 1 kb	30
TPC-E	Carga de trabalho OLTP (corretor da bolsa)	Clientes ativos = 200 000; dias de comércio = 7; Terminais = 128; *innodb-thread* concorrentes = 8; *innodb*-arquivo-io-*threads = 4*	155
BLAST	Busca de similaridade de sequência	Instâncias = 16; *Threads* por instância = 16; Tarefa=*blastn;* Questionário por instância = 16; Bancos de dados de nucleotídeo pré-formatado de NCBI (refseq-genomic, env-nt, nt); Alinhamentos = 128; sequências meta = 5000	20

que o pico de utilização da memória do Dedup é muito alto para arquivos grandes, pois as filas entre os estágios ficam muito grandes. Para eliminar as filas rapidamente, é necessário um elevado número de *threads* nos estágios intermediários. A utilização de memória não é um problema quando se trata de pequenos arquivos, porque há menos estados intermediários a serem mantidos. O Dedup é inteiramente dominado pelo tempo do usuário.

O Madrepuce é um modelo de programação cada vez mais usado para o desenvolvimento de aplicações centradas em dados. Usamos uma biblioteca de programação Madrepuce para nós únicos, chamada Metis, que é parte do conjunto de referência Mosbench [5]. A Metis mapeia o arquivo de entrada na memória e atribui uma parte do arquivo para cada um dos *threads* de mapa. Usamos o Metis para contar palavras em um arquivo. Uma única instância do Metis não gera muita E/S e é em grande parte dominada pelo tempo do usuário. Rodamos múltiplas instâncias de Metis e atribuímos um arquivo diferente para cada instância.

Os armazenamentos de dados NoSQL estão ficando populares para servir dados de uma forma escalável. O HBase é um sistema de serviço de dados que faz parte da estrutura Hadoop. Usamos a estrutura YCSB [6] do Yahoo para testar o HBase. Primeiro, construímos um banco de dados usando o gerador de carga YCSB (usando uma carga de trabalho que executa apenas operações de inserção). Em seguida, rodamos uma carga de trabalho que executa 70% das operações de leitura e 30% de operações de atualização. Reservamos 3 Gb de memória física para a máquina virtual Java (JVM). O HBase tem um tempo ocioso alto, embora haja uma igual quantidade de tempo de usuário e de sistema.

O armazenamento de dados *key-value* (para cada valor há uma chave correspondente) continua sendo uma escolha popular para servir dados em data centers. O BDB é uma biblioteca que suporta a construção de dispositivos de armazenamento de dados com base em pares *key-value* (chave-valor).

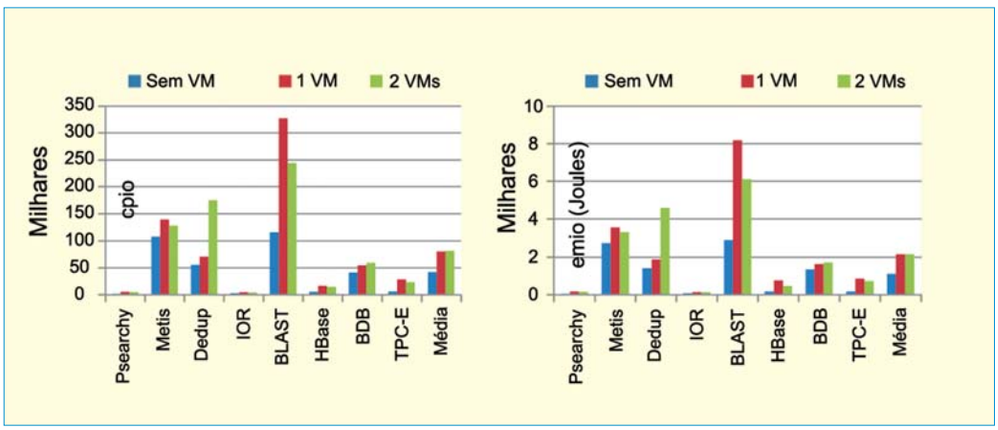

Fig. 4 - cpio (esquerda) e emio (direita) com e sem virtualização em SSDs

A metodologia de avaliação de BDB é similar àquela que usamos para HBase. Uma vez que o BDB é um dispositivo de armazenamento de dados embutido, os clientes YCSB e o código BDB compartilham o mesmo espaço de endereço de processamento. Portanto, reservamos 6 Gb de memória física para a JVM. Configuramos o YCSB para usar 3 Gb para os clientes YCSB e 3 Gb para BDB. O BDB é dominado por tempo de usuário, mas há um considerável tempo de sistema.

O processamento de transações on-line (OLTP - online transaction processing) é uma importante classe de cargas de trabalho para data centers. Usamos TPC-E para avaliar uma carga de trabalho OLTP. O TPC-E modela as transações que acontecem numa corretora de ações. Rodamos o TPC-E usando o sistema de banco de dados MySQL e especificamos os parâmetros de tempo de execução que resultam em alta concorrência. Observamos que usar o servidor de banco de dados MySQL resulta em alto tempo ocioso para TPC-C e TPC-E, seja devido à não escalabilidade para múltiplos núcleos ou ociosidade devido à sincronização entre um grande número de *threads*.

A genômica comparativa mobiliza uma enorme quantidade de dados genômicos graças aos avanços da tecnologia de sequenciamento. Usamos BLAST [2] para busca de similaridade de sequência nucleotídeo-nucleotídeo. Rodamos múltiplas instâncias de BLAST, cada qual executando um conjunto diferente de consultas em bancos de dados separados. Usamos sequências de consultas aleatórias de 5 kB, que é um caso comum em pesquisas de homologias proteoma/genoma. O BLAST é intensivo quanto a E/S e o tempo de execução é dominado pelo tempo do usuário.

Plataformas de avaliação e o impacto dos subsistemas de armazenamento

Caracterizamos o comportamento das cargas de trabalho na tabela I usando um servidor baseado em disco e um baseado em SSD – solid state drive. Nossa principal finalidade em usar dois servidores diferentes é observar como a divisão da execução de cargas de trabalho é efetuada pelo hardware de armazenamento. Os principais recursos das duas máquinas que usamos para a avaliação estão na tabela II.

Para testes com virtualização, usamos a infraestrutura de virtualização para kernel Linux (KVM). Notamos que a KVM requer que pelo menos um núcleo e até dois núcleos sejam reservados para que a KVM destinada a cargas de trabalho rode de forma adequada. Realizamos testes de virtualização usando apenas oito processadores (lógicos) e 8 Gb de DRAM. Portanto, deixamos oito processadores lógicos para KVM e 4 Gb de DRAM para o kernel (núcleo) de hospedagem. Quando executamos duas VMs, alocamos quatro processadores lógicos e 4 Gb de DRAM para cada instância de VM. Da mesma forma, dividimos nosso subsistema de armazenamento em SSDs para dois dispositivos RAID 0, cada qual com 12 SSDs, e atribuímos um para cada das duas instâncias de VM.

Primeiro, observamos que, em média, o tempo do sistema com disco físico é de apenas 2% do tempo de execução total, em comparação com os 26% em SSDS.

Entretanto, o resultado mais sutil é que os ciclos gastos por I/O (cpio) e, em particular, os ciclos do sistema por E/S, são quase os mesmos nas duas máquinas. Portanto, a eficiência da pilha do sistema, ou seja, as camadas OS e outras abaixo da aplicação do usuário, ainda é relevante para ambas as máquinas. No entanto, para compreender as sobrecargas introduzidas pelo acréscimo da camada VM, relatamos os resultados usando SSDs. E também, realizamos os testes de escalabilidade e relatamos os resultados de SSDs. Em essência, as tendências da escalabilidade são mais fáceis de identificar e projetar quando a E/S não é o gargalo.

A estimativa da potência dos servidores de faixa média, hoje, é de cerca de 400 W, valor que usamos para nossos cálculos de energia. Como o cpio é bastante independente de um servidor particular, a potência específica que usamos para calcular eio (e emio) não é importante para a comparação das várias camadas do sistema.

A figura 1 mostra a divisão do tempo de execução em termos de tempo do usuário, do sistema, ocioso e iowait. Como esperado, a máquina com subsistema de armazenamento baseado em disco tem um alto tempo iowait. Em média, o tempo iowait com disco é 42% do tempo de execução total, comparado com os 10% em SSDs. Entretanto, notamos que o tempo ocioso também é alto na máquina baseada em disco. Em média, a porcentagem de tempo ocioso é de 30% com disco, comparados com apenas 7% em SSDs. Isso é porque, normalmente, nas aplicações de hoje, há *threads* cuja principal tarefa é executar E/S e distribuir trabalho ou fornecer dados a um grande número de *threads* que executam a principal funcionalidade da aplicação. Se o subsistema E/S é lento, os *threads* que executam E/S têm menos trabalho a distribuir a outros *threads* e, portanto, todo o sistema é ineficiente ou há mais tempo ocioso.

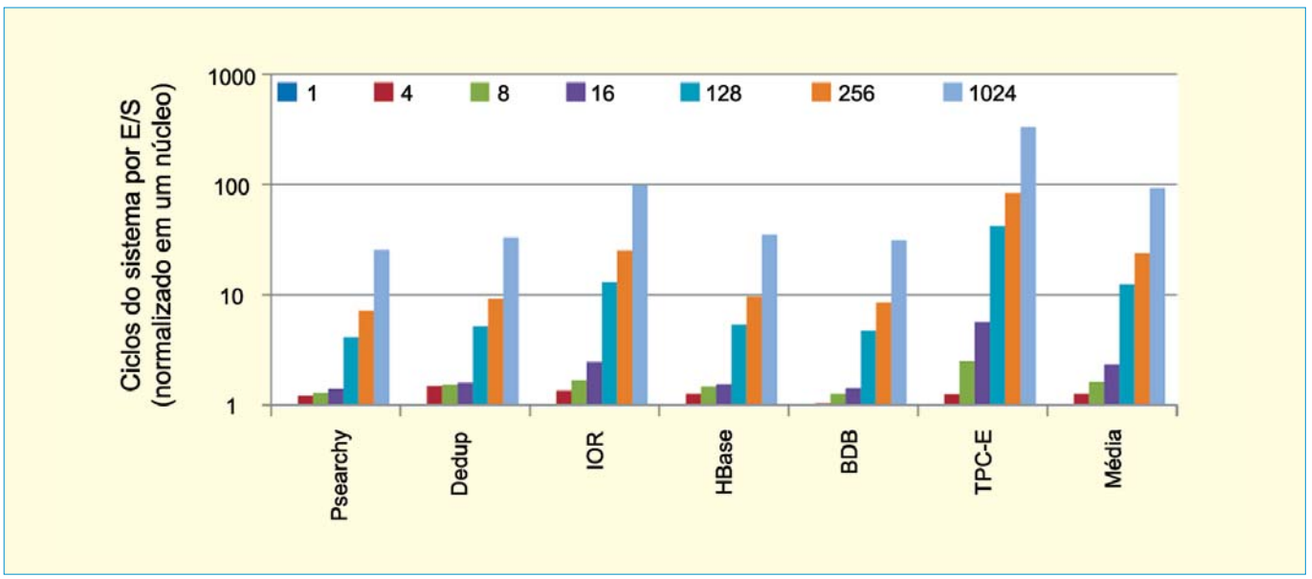

Fig. 5 - Aumento em ciclos do sistema por operação E/S a partir de 1 até muitos núcleos

Essa observação é de particular interesse para infraestruturas centradas em dados, uma vez que nos servidores de hoje os períodos de tempo ocioso e iowait consomem até 70% do pico de energia. Com isso, os subsistemas baseados em disco não apenas terão um tempo de resposta mais lento, mas também ineficiente no processamento de E/S, em termos de energia. A figura 2 mostra o emio de ambas as máquinas. Mostramos o emio de ambas as máquinas, supondo uma potência ociosa (IP - idle power) de 0% e 70% do pico de potência. Primeiro, notamos que, em média, com IP = 0, o emio com disco é 58% maior do que o de SSDs. Entretanto, com IP = 0,7, o emio com disco é seis vezes o de SSDs. Concluímos que os servidores de hoje em dia, em que a IP está mais próxima de 0,7, são ineficientes em termos de energia para operar aplicações com o uso de subsistemas de armazenamento com baixo *throughput*. No entanto, uma vez que a potência ociosa seja reduzida para aproximadamente 0% do pico, a diferença em termos de ineficiência energética será bastante reduzida.

Com o uso de disco, os ciclos de tempo ocioso e iowait têm um custo significativo em termos de consumo de energia. Em média, uma IP de 0,7 comparada a 0 resulta em um aumento de 4x em emio com disco. Nesse trabalho, fizemos um esforço para executar todas as cargas de trabalho com o máximo de simultaneidade, para maximizar a utilização do servidor. Em certos casos, em que a interferência e a contenção não significam um problema, rodamos múltiplas instâncias da mesma aplicação. Portanto, os ciclos ociosos das cargas de trabalho avaliadas nesse trabalho são um comportamento da aplicação. À parte os ciclos ociosos excessivos com disco, esse comportamento tem implicações também para a máquina de SSDs. Por exemplo, tanto o HBase quanto BDB não utilizam integralmente os recursos do servidor em SSDs e com disco e, portanto, há uma notável diferença entre o emio com IP de 0 e com IP de 0,7.

Quantificação de sobrecarga de várias camadas

Aqui comparamos o emio do componente usuário sozinho, o de usuário e do componente sistema juntos, o da mesma aplicação rodando dentro de uma VM e o de duas instâncias da mesma aplicação rodando em duas VMs separadas. Todos os resultados são mostrados para uma IP de 0,7.

Primeiro, a figura 3 compara o emio com e sem sobrecarga de sistema para cargas de trabalho com tempo de

sistema significativo (10% ou mais, em SSDs). Notamos que, com disco, não há qualquer diferença de emio com ou sem sobrecarga de sistema. Isso é porque o tempo de execução é dominado pelo tempo ocioso ou por iowait. Entretanto, em SSDs, a introdução da camada de sistema resulta em um aumento de 60% do emio (em média, cinco aplicações). A sobrecarga máxima é do Psearchy (250%). Note-se que o Psearchy é uma aplicação simples, mas acaba por consumir uma grande quantidade de energia, principalmente para acessar os arquivos para a construção de índices. Embora não seja realista supor que as sobrecargas no nível de sistema serão completamente eliminadas, o trabalho mostra a faixa em que as pilhas modernas poderiam ser melhoradas quanto à eficiência energética.

A seguir, discutimos e comparamos os resultados com a virtualização. A figura 4 mostra a sobrecarga que se deve à virtualização, tanto em termos de cpio quanto de emio. Na figura, mostramos os resultados com uma instância de carga de trabalho sem virtualização, com a mesma carga de trabalho rodando dentro de uma VM e com duas instâncias de carga de trabalho cada qual em uma VM separada.

Em termos de desempenho, vemos que, em média, o cpio aumenta em 150%. Note-se que três cargas de trabalho que incluem Dedup, Metis e BDB têm baixa sobrecarga de até 30%, devido à virtualização, em comparação com outras cargas de trabalho. Note-se que a porcentagem do tempo do sistema em relação ao tempo de execução total, com as três cargas de trabalho, é de 0,6%, 7,2% e 10%, respectivamente. Há uma possibilidade de que essas aplicações operem em grande parte a partir dos caches mantidos pela aplicação ou pelo OS de hospedeiros. Nas demais cargas de trabalho, a sobrecarga devida à virtualização é de até 300%.

Na maioria das cargas de trabalho, há alguma redução da sobrecarga da virtualização, quando se usam duas VMs para rodar duas instâncias da mesma carga de trabalho. Em particular, para IOR e BLAST, há uma redução de 16% e 25% em cpio, quando duas instâncias de VM são rodadas, em comparação com a execução de uma instância de VM. Entretanto, observamos que para Dedup e BDB, a sobrecarga devido à execução de duas VMs é realmente aumentada. Em particular, para Dedup, há um aumento de 200% da sobrecarga de VM com duas VMs, em comparação com os 28% de sobrecarga com uma VM.

Finalmente, notamos uma tendência semelhante em termos das sobrecargas de energia da virtualização. Em média, em todas as cargas de trabalho, a virtualização resulta num aumento de consumo de energia de 180%. A sobrecarga é particularmente adversa em duas importantes cargas de trabalho dos data centers de hoje, incluindo HBase e TPC-E. Nas duas cargas de trabalho, a sobrecarga é de 350% e 400%, respectivamente.

Tab. II - Resumo de parâmetros da máquina

DISKS	SSDs
2 Intel Xeon E5620 (núcleo quádruplo)	2 Intel Xeon E5405 (núcleo quádruplo)
Sem *Hyper-threading*	2 hardware *thread* por núcleo
8 Gb RAM; 1 controlador de armazenamento	12 Gb RAM; 4 controladores de armazenamento
XFS sobre hardware RAID 0 (8 discos)	XFS sobre software RAID 0 (24 SSDs)
Throughput de armazenamento = 1 Gbit/s	*Throughput* de armazenamento = 6 Gbit/s
Distribuição CentOS; 2.6.18 *kernel*	Distribuição CentOS; 2.6.32 *kernel*

Escalabilidade da camada OS

Idealmente, com o crescente número de núcleos, a aplicação deve executar, proporcionalmente, mais E/S, assim resultando em cpio constante.

Entretanto, observamos que isso não é verdade com as pilhas dos sistemas atuais. Nesse trabalho, estamos principalmente interessados no componente sistema de cpio. Mostramos o aumento dos ciclos do sistema por operação de E/S com o número de núcleos na figura 5.

Mostramos os valores medidos com 1, 4, 8 e 16 núcleos. Em seguida, projetamos os resultados para até 1000 núcleos, usando um modelo linear. Apresentamos o aumento de ciclos do sistema normalizado para os ciclos medidos com um núcleo.

Os ciclos de sistema aumentam em todas as aplicações mostradas na figura 5. As mesmas aplicações executadas com um processador de muitos núcleos, com 1024 núcleos, gastarão, em média 90 vezes mais ciclos de sistema por operação E/S. Isso indica que a atual camada de sistema é ineficiente e são necessárias melhorias para otimizar as aplicações que usam intensamente a camada de sistema. Notamos que, em particular nas cargas de trabalho OLTP mais recentes, o aumento de ciclos do sistema com o aumento do número de núcleos é dramático. O TPC-E depende bastante dos serviços OS, gastando até 36% de seu tempo de execução no OS com 16 núcleos.

Os ciclos de sistema consumidos por uma aplicação traduzem-se diretamente em sobrecargas de energia. Portanto, a não escalabilidade da camada OS tem implicações adversas para o consumo de energia em data centers.

Trabalho correlato

O interesse em pesquisar questões relativas à ineficiência de pilhas de software em aplicações centradas em dados está crescendo. [17] discute tendências na construção de aplicações de data centers a partir de componentes existentes, que levam a grandes ineficiências. Trabalhos recentes apontam que as ineficiências da pilha de software têm impacto sobre a eficiência energética de infraestruturas de data centers [3, 14].

A metodologia dominante, até o momento, do lado da redução de energia, tem sido para construir modelos de potência fazendo primeiro medições de um subconjunto do espaço de projeto. Usando essa abordagem, há uma forte evidência em [9] e [20] de uma relação direta entre o consumo de energia em grupos que rodam típicas cargas de trabalho centradas em dados e a utilização da CPU e o uso da memória física. As medições em [9] sugerem ainda que a CPU e a memória física são as que mais contribuem para a potência. As atuais técnicas de modelagem são basante discutidas em [19, 18, 8].

O programa ENERGY Star da Agência de Proteção Ambiental dos EUA estima os requisitos de infraestrutura e o consumo de energia de data centers em 2020, com base no crescimento do mercado [1]. Eles também sugerem otimizações que provavelmente reduzirão os orçamentos energéticos, reduzindo a potência de servidores usados em data centers.

Há trabalhos mais antigos sobre a análise de desempenho das modernas camadas de sistema operacional. Mais recentemente, [5] identificou muitos gargalos no kernel Linux quando muitos núcleos são utilizados. Descreve-se uma análise de escalabilidade similar para muitos domínios importantes de aplicação em [7, 21]. Finalmente, há trabalhos recentes sobre a redução de sobrecargas devido à adição da camada de virtualização em cargas de trabalho com grande intensidade de E/S. Os autores em [13] propõem o Split-X, que otimiza os cruzamentos de OS hóspede-hospedeiros, enquanto [11] propõe ELI para a otimização e interrupção de via sob as máquinas virtuais.

Conclusões

Neste artigo, usamos um conjunto de aplicações com intensidade de uso de E/S para quantificar a sobrecarga do OS e das máquinas virtuais ao executarem E/S relativas ao processamento de aplicações. O OS pode resultar em um aumento de 60% do consumo de energia. O acréscimo de máquinas virtuais resulta numa sobrecarga energética adicional, por operação E/S, de 150%. Finalmente, observando um crescente número de núcleos, verificamos que a sobrecarga por E/S, no OS, aumenta em 90 vezes, ao fazermos a projeção para 1000 núcleos, usando um modelo linear simples.

Agradecimentos

Os autores agradecem à Comunidade Europeia, sob o 7th Framework Programs e IOLANES (FP7-ICT-248615), HiPEAC2 (FP7-ICT-217068) e SCALUS (FP7-PEOPLE-ITN-2008-238808). O autor Angelos Bilas também da Universidade de Creta, Grécia.

REFERÊNCIAS

[1] U. E. P. Agency. *Report to congress on server and data center energy efficiency*. www.energystar.gov.

[2] S. Altschul, W. Gish, W. Miller, E. Myers e D. Lipman: *Basic local alignment search tool.* Journal of Molecular Biology, 215:403–410, 1990.

[3] E. Anderson e J. Tucek: *Efficiency matters!* SIGOPS Oper. Syst. Rev., 44(1):40–45, 2010.

[4] C. Bienia, S. Kumar, J. P. Singh e K. Li: *The parsec benchmark suite: characterization and architectural implications.* 17th International Conference on Parallel Architectures and Compilation Techniques, EUA. 2008.

[5] S. Boyd-Wickizer, A. T. Clements, Y. Mao, A. Pesterev,

REFERÊNCIAS

M. F. Kaashoek, R. Morris e N. Zeldovich: *An analysis of linux scalability to many cores*. 9th USENIX Conference on Operating Systems Design and Implementation, OSDI'10. EUA. 2010.
[6] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan e R. Sears. *Benchmarking cloud serving systems with ycsb*. 1st ACM symposium on Cloud computing, SoCC '10. EUA, 2010.
[7] Y. Cui, Y. Chen e Y. Shi: *Scaling oltp applications on commodity multi-core platforms*. Performance Analysis of Systems Software (ISPASS), 2010 IEEE International Symposium.
[8] D. Economou, S. Rivoire e C. Kozyrakis: *Full-system power analysis and modeling for server environments*. Workshop on Modeling Benchmarking and Simulation. MOBS, 2006.
[9] X. Fan, W.-D. Weber e L. A. Barroso: *Power provisioning for a warehouse-sized computer*. ISCA '07: 34th Annual International Symposium on Computer Architecture. EUA, 2007.
[10] J.F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. Mcarthur, S. Minton, I. Xheneti, A. Toncheva e A. Manfrediz: *IDC - The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010*.
[11] A. Gordon, N. Amit, N. Har'El, M. Ben-Yehuda, A. Landau, D. Tsafrir e A. Schuster. Eli: *Bare-metal performance for i/o virtualization*. 2012.
[12] R. Joshi. Data-Centric Architecture: *A Model for the Era of Big Data*.
[13] A. Landau, M. Ben-Yehuda e A. Gordon: *Splitx: split guest/hypervisor execution on multi-core*. 3rd conference on I/O virtualization, WIOV'11. EUA, 2011.
[14] J. Leverich e C. Kozyrakis: *On the energy (in)efficiency of hadoop clusters*. SIGOPS Oper. Syst. Rev., 44(1):61–65, 2010.
[15] J. Li, B. T. Loo, J. M. Hellerstein e M. F. Kaashoek et al.: *On the feasibility of peer-to-peer web indexing and search*. IPTPS.03. 2003.
[16] llnl.gov: *ASC Sequoia Benchmark Codes*. https://asc.llnl.gov.
[17] N. Mitchell: *The big pileup. In Performance Analysis of Systems Software (ISPASS)*, 2010 IEEE International Symposium on.
[18] S. Rivoire, P. Ranganathan e C. Kozyrakis: *A comparison of high-level full-system power models*. HotPower'08: Conference on Power aware computing and systems, EUA, 2008.
[19] S. Rivoire, P. Ranganathan e C. Kozyrakis: A comparison of high-level full-system power models. HotPower'08: Power aware computing and systems. EUA, 2008.
[20] A. Vasan, A. Sivasubramaniam, V. Shimpi, T. Sivabalan e R. Subbiah: *Worth their watts? - an empirical study of datacenter servers*. 2010.
[21] B. Veal e A. Foong: *Performance scalability of a multi-core web server*. 3rd ACM/IEEE Symposium on Architecture fornetworking and communications systems, ANCS '07, EUA, 2007.