Reconfiguração Automática de Quantum de Escalonamento no Xen

Juliano Potrich[1], **Fábio Diniz Rossi**[2]

[1]Pontifícia Universidade Católica do Rio Grande do Sul - Faculdade de Informática
Av. Ipiranga, 6681 - Prédio 32 - CEP 90619-900 - Porto Alegre - Brasil

[2]Instituto Federal Farroupilha - Campus Alegrete
RS 377 - Km 27 - Caixa Postal 118 - Alegrete - RS - Brasil

jpotrich@gmail.com, fdrossi@al.iffarroupilha.edu.br

1. Introdução

Nos últimos anos, a capacidade de processamento dos computadores tem aumentado consideravelmente. Entretanto, toda a capacidade oferecida não vem sendo utilizada ao máximo. Há situações onde aplicações poderiam ser executadas de uma maneira mais eficiente potencializando o uso de CPU (*Central Processor Unit*). Uma das soluções é o uso da virtualização, que está ganhando cada vez mais destaque, e sendo utilizada com resultados bastante satisfatórios. Um dos monitores de máquinas virtuais (VMM) com grande expressão na academia é o Xen. Seu escalonador padrão mostra algumas limitações, portanto este trabalho mostra uma modificação no escalonador, sua análise de desempenho e trabalhos futuros.

Este artigo está organizado em 4 seções. Uma introdução sobre a pesquisa é apresentada na primeira seção. A segunda seção mostra o VMM Xen, com ênfase na estrutura interna de seu escalonador padrão (*SMP Credit*). Na terceira seção veremos as modificações realizadas no código-fonte do escalonador. Em seguida, a quarta seção apresenta avaliações de desempenho do escalonador com as modificações propostas e nossas conclusões.

2. Escalonador do Xen

O Xen foi desenvolvido pelo *Systems Research Group* da Universidade de Cambridge [Barham et al. 2003], e permite compartilhar uma simples máquina para vários clientes rodando sistemas operacionais e programas. Este VMM utiliza o conceito de paravirtualização o que possibilita ao sistema operacional que executa sobre uma máquina virtual (VM) ter a ilusão de estar sendo executado diretamente sobre o *hardware*. O Xen se encarrega de organizar a maioria das requisições feitas pelas VMs e repassá-las ao domínio0 (sistema hospedeiro).

O *SMP Credit* [Antoniou and Stavrakakis 2002] é um escalonador construído para manter um trabalho justo para todos os processadores em máquinas SMP no Xen. A cada domínio convidado são atribuídos dois valores: um peso e um limite. Este escalonador provê, de forma automática, balanceamento de carga entre as VCPUs (CPUs virtuais) dos domínios convidados, utilizando todas as CPUs de uma máquina SMP. Uma vantagem desse escalonador, é que o administrador não necessita referenciar cada VCPU a cada CPU, o escalonador já trabalha dessa maneira, associando uma VCPU a uma CPU.

Cada CPU controla uma fila local do funcionamento de VCPUs em execução. Esta fila é classificada pela prioridade de cada VCPU. Uma prioridade de VCPUs pode ser um dos dois valores: *over ou under*, que representam se a VCPU excedeu ou não a sua fatia de acesso justa ao recurso do processador. Enquanto uma VCPU executa, consome créditos. Assim, frequentemente é recalculada a contabilidade de quantos créditos cada VM ativa ganhou. Os créditos negativos implicam em uma prioridade *over*.

Até que uma VCPU consuma todos seus créditos, sua prioridade estará *under*. Quando uma CPU não encontra uma VCPU de prioridade *under* em sua fila local de funcionamento, buscará em outra CPU por uma VCPU de prioridade *under*. Assim, tenta garantir balaceamento de carga, onde cada VM recebe sua parte justa de recursos do processador. Antes que um processador fique inativo, olhará em outro processador para encontrar uma VCPU executável. Isto garante que nenhum processador trabalhe lentamente quando existem execuções a cumprir no sistema. Portanto, o escalonador tem a capacidade de quando uma CPU estiver parada, buscar VCPUs de outras CPUs para execução. O *SMP Credit* prima por distribuir tarefas de forma justa as fatias de CPU para os domínios virtuais, porém dependendo do tipo de processos que estiverem rodando nas VMs, esse recurso não é distribuído de forma justa. O principal problema está em que o seu tempo de processamento para as VMs, que é estático, ou seja 30 milisegundos.

3. Otimização do Código-Fonte

Na Figura 1 podemos observar como funciona o escalonador *SMP Credit*, e quais etapas são percorridas para realocar suas novas configurações.

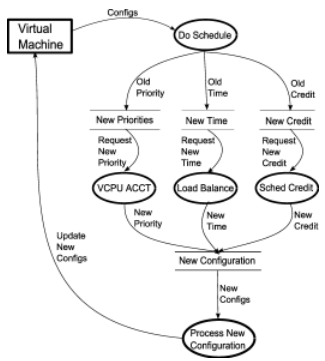

Figura 1. Xen: Mudanças de Contexto

Primeiramente a VM que será escalonada contém informações como créditos, prioridade do VCPU, e tempo de acesso ao CPU (*configs*), que passa ao *Do Schedule*, realizando o escalonamento. Estas informações são delegadas para o *New Priorities, New Time, New Credit*, que verificam se essas informações estão de acordo com as prioridades, tempo, e créditos do escalonador, e se é necessário mais processamento para a VM.

Havendo a necessidade de maior fatia de processamento, é feita uma requisição de novos parâmetros (prioridade, tempo e créditos) através dos processos *VCPU ACCT*

(aumenta a prioridade do VCPU), *Load Balance* que determina um tempo de acordo com a prioridade, e o *SCHED CREDIT* que realiza a contagem de créditos, assim realizando uma nova configuração (*New Configuration*).

Com essas novas configurações, é executado um processo de verificação (*Process New Configuration*), que verifica se esta nova configuração irá aumentar o desempenho, aplicando a mesma para a VM (*update new configs*). Para solucionar o problema de tempo de acesso ao CPU, propomos a redefinição de alguns parâmetros, como o tempo de escalonamento que era estático com 30 milisegundos para cada associação VCPU a domínio. Se o quantum for muito grande, o comportamento será igual a um escalonador *first-in-first-out*, por outro lado, se o quantum for muito pequeno, o desempenho é comprometido pelo número excessivo de preempções e consequentes atrasos para a troca de contexto.

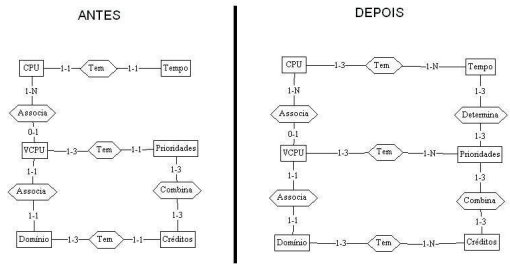

Figura 2. Xen: Prioridades

Na Figura 2, representado no diagrama E-R, vemos a estrutura desse relacionamento antes e depois da solução. Como pode-se observar todo o domínio possui uma VCPU, e todo VCPU possui CPU. Sabemos que cada domínio possui créditos que liga com as prioridades dos VCPU, por exemplo, dependendo do número de créditos, se altera a prioridade do VCPU, mas continuamos sem nenhuma conexão com o tempo de escalonamento, que permanece constante em 30 milisegundos. Ainda na Figura 2, depois da implementação da solução podemos perceber que há uma relação entre prioridades do VCPU com tempo de escalonamento, pois dependendo da prioridade dos domínios, será alterado o tempo de acesso ao CPU.

Agora são considerados três tipos de tempos, implementados através das seguintes definições: o normal, tempo padrão definido pelo *SMP Credit* (30 milisegundos de acesso ao CPU); o máximo, que determina 30% a mais tempo de acesso ao CPU que o definido pelo *SMP Credit*; o mínimo, que lhe concede 30% a menos de tempo de acesso ao CPU que o definido pelo *SMP Credit*. Para haver uma ligação entre tempo de escalonamento e a implementação de níveis de serviços, alteramos a contagem de créditos. Por exemplo, se existir uma prioridade máxima de tempo para alguma VM, então deve haver uma contagem de créditos máxima, assim potencializando ainda mais o desempenho das VMs. Para processos que se utilizam intensivamente da CPU, essa metodologia se torna interessante, pois notamos em nossas avaliações que um ambiente com processos *CPU-Bound* tem mais desempenho com um quantum maior.

4. Análises e Conclusões

Foram realizados testes (execução de *benchmarks*) sobre as VMs (200 execuções com o escalonador modificado, os 10% maiores e os 10% menores resultados foram excluídos da média final para manter um intervalo de confiança), proporcionando formas padronizadas de avaliação do desempenho de sistemas, retirando estatísticas do seu funcionamento. O *benchmark* escolhido foi o Unixbench, utilizado para encontrar gargalos em partes específicas dos subsistemas do *kernel*.

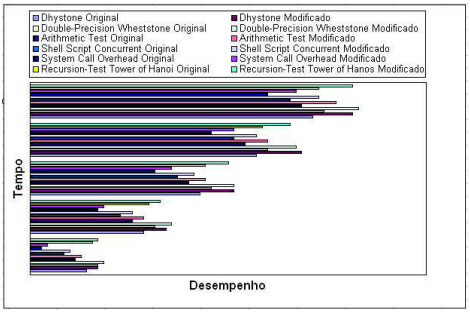

Figura 3. Avaliações de Desempenho

Para tornar as VMs diferentes em relação a tempo de acesso ao CPU, a implementação propôs novas prioridades ao sistema, adicionando definições máxima e mínima na contagem de créditos e de tempo. As definições implementadas criam tempos dinâmicos de acesso ao CPU, pois sempre que alguma prioridade for alterada, ou existir necessidade de maior desempenho, seu tempo de acesso modificará.

Conforme a Análise dos resultados, utilizando o *benchmark* Unixbench, podemos observar na Figura 3 uma melhora significativa em média de 12% no desempenho do escalonamento (nas métricas avaliadas: *Dhrystone*: iterações de um laço/segundo, *Whetstone*: operações de ponto flutuante, *Arithmetic Test*: testes aritméticos, *Shell Script Concurrent*: concorrência entre processos , *System Call Overhead*: latência das chamadas de sistema e *Recursion Test - Tower of Hanoi*: recursividade) com as modificações propostas para processos *CPU-Bound*.

5. Agradecimentos

Avelino Francisco Zorzo - FACIN/PUCRS e HP Brasil.

Referências

Antoniou, Z. and Stavrakakis, I. (2002). An efficient deadline-credit-based transport scheme for prerecorded semisoft continuous media applications. *IEEE/ACM Transactions on Networking*, 10(5):630–643.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, Bolton Landing, NY, USA. ACM Press.