Projeto 2 - Tradução de memória virtual para física

Alceu Emanuel Bissoto[1]

[1]Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

Abstract. *This work evaluates the virtual to physical memory translation. To be able to do that, the main components such as the TLB and different cache levels have been simulated using the instrumentation software PIN made by Intel, which helps us to measure the implications of changing the TLB's page size. Three SPEC2006 benchmarks have been used and a toy benchmark thas been created, in order to test it in different scenarios. The result of these tests showed that changing the TLB's page size to 4MB greatly reduced the amount of memory access at most of the cases.*

Resumo. *Neste trabalho é descrito o processo de avaliação para o processo de tradução de memória virtual para memória física. Para tal, foram simulados através do* PIN *da [Intel] componentes essenciais deste processo como a TLB e os diferentes níveis de cache, de modo a medir as implicações da mudança do tamanho da página da TLB de 4kB para 4MB. Foram testados três benchmarks do [SPEC] e um toy benchmark que faz escritas e leituras em diferentes posições da memória. O resultado obtido mostra que, no ambiente isolado utilizando apenas um processo, como foi nesse experimento, a página de 4MB mostrou diminuir muito na grande maioria dos casos a quantidade de acessos a memória.*

1. Introdução

O processo de tradução de memória virtual é parte do pipeline de processadores que utilizam-se desta para obter benefícios de segurança e desfragmentação, além de isolar diferentes processos que não deveriam criar interferência entre si.
Este processo é feito pela estrutura chamada de tabela de páginas, e pode conter vários níveis de modo a economizar espaço em memória. Há ainda uma *cache* para a page table chamada de *Translation Lookaside Buffer* (TLB) que torna a tradução para endereços recentemente utilizados muito mais rápida.
Neste trabalho, o *software* de instrumentação *PIN* da *Intel* foi utilizado como simulador de cache de modo que fosse verificado a quantidade de *misses* na TLB e na cache L3, para que a quantidade de acessos a memória pudesse ser estimado. Desta maneira, foram reunidos a quantidade de *misses* na TLB de instrução e de dados, assim como a quantidade de acessos a tabela de páginas e também o acesso total à memória. Com estas informações, foi possível verificar o impacto (em número de acessos a memória) que um *miss* na TLB causa.
Foram avaliadas TLBs com tamanho de página de 4kb e 4Mb, comparando o resultado obtido.

2. SPEC benchmark

A primeira análise a ser feita foi utilizando o *SPEC2006*, instrumentando três de seus programas. Foram analisados os seguintes *benchmarks*:

- 429.mcf - Otimização combinatorial.
- 456.hmmer - Busca de sequência de genes.
- 471.omnetpp - Simulação de eventos discretos.

Nas figuras 1, 2 e 3 é possível ver a diferença causada para os *benchmarks 429.mcf, 456.hmmer, e 471.omnetpp* respectivamente.
É notável que em todos os casos, a mudança do tamanho da página da TLB de 4kb para 4Mb resultou em uma diminuição considerável no número de acessos a memória. Após o aumento do tamanho da página, a quantidade de acessos à memória foi reduzida apenas aos acessos decorrentes de *misses* no último nível de cache, L3 neste caso.

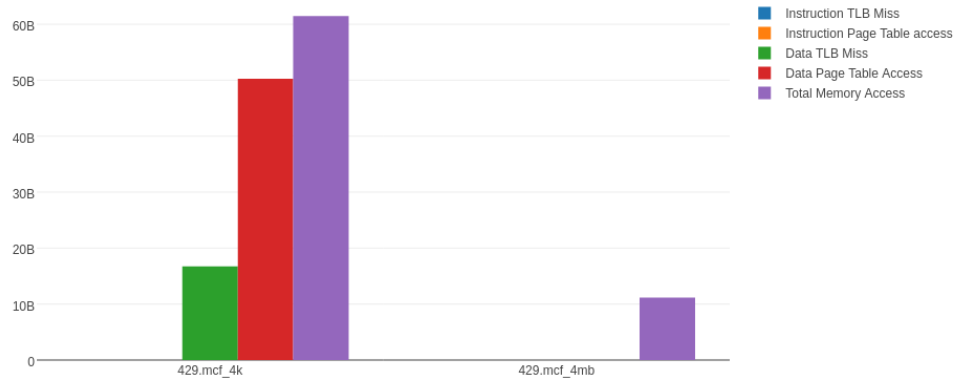

Figura 1. Comparação utilizando tabela de páginas de 4kb e 4mb para o programa 429 do *SPEC2006*

3. Toy Benchmark

Um *toy benchmark* foi criado de modo a explorar o comportamento da TLB para diferentes tamanhos de página. Foi implementado um código em C++ que cria um vetor que aloca muita memória, e através das iterações, provoca repetidas escritas e leituras no mesmo. O intuito é provocar vários misses na TLB, já que as escritas e leituras acontecem em posições distantes no vetor, forçando que um novo bloco de memória seja carregado, e uma nova entrada na TLB seja adicionada várias vezes durante a execução.
Temos abaixo um trecho do código utilizado. Variou-se neste estudo os parâmetros *page_size* e *k*. O número de entradas da TLB foi mantido em 512 de modo a coincidir com o utilizado no simulador de cache no PIN.

```
1    std::vector<int> vetor(tlb_size*page_size*k);
2    for (int h=0; h<10000; h++) {
3        for (int j=0; j<tlb_size*k; j++) {
4            vetor[j*page_size]++;
5        }
6    }
```

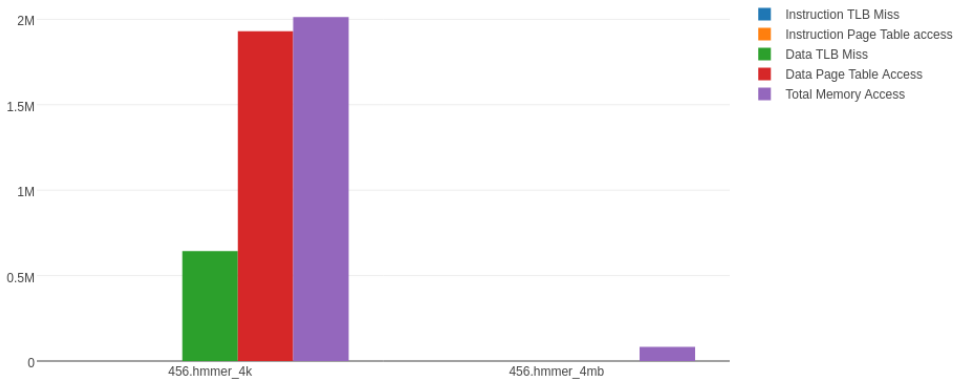

Figura 2. Comparação utilizando tabela de páginas de 4kb e 4mb para o programa 456 do *SPEC2006*

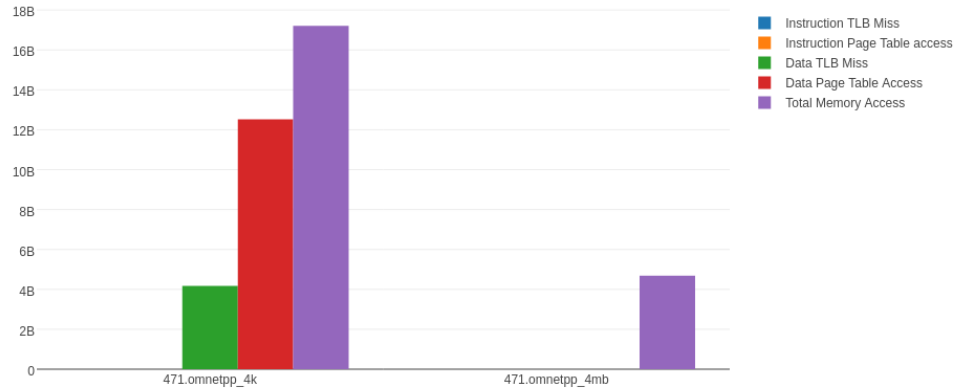

Figura 3. Comparação utilizando tabela de páginas de 4kb e 4mb para o programa 471 do *SPEC2006*

Foram feitos dois experimentos: variando k, e o tamanho da página.
Primeiramente, manteve-se *page_size* em 4096, e variou-se *k* entre valores de 2 a 20, utilizando o PIN como simulador de cache, de modo a reunir informações sobre a execução. Temos na figura 4 o resultado para 5 valores de *k*: 2, 6, 10, 15 e 19.

É possível verificar que o número de acessos cresce linearmente conforme k aumenta. O principal resultado é que utilizando estes parâmetros, a TLB de páginas de 4MB se sai muito melhor, reduzindo a quantidade de *misses* na TLB para valores muito baixos.
Utilizando páginas de 4MB na TLB, o total de acessos a memória se resume ao processador buscando dados que não foram encontrados em nenhum dos níveis da cache (*miss* na

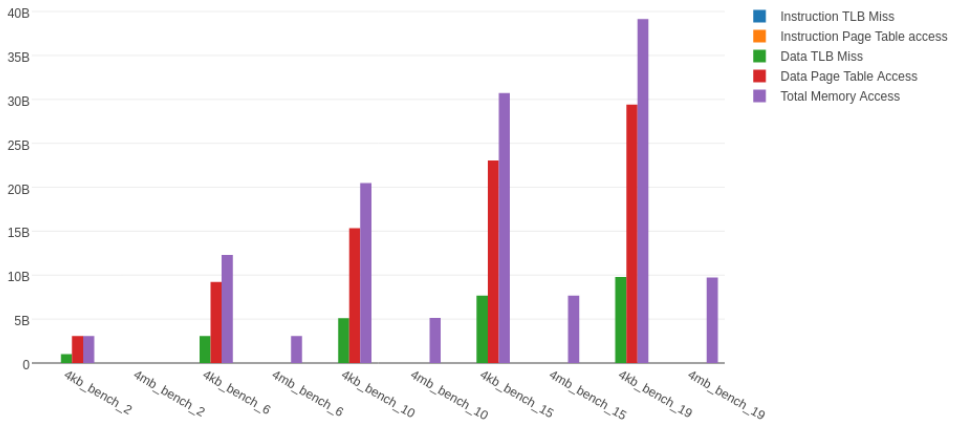

Figura 4. Comparação utilizando tabela de páginas de 4kb e 4mb para diferentes entradas do *toy benchmark* criado.

L3), já que quase não há *misses* na TLB. Por outro lado, quando são utilizadas páginas de 4KB na TLB, a grande maioria dos acessos são decorrentes justamente de *misses* na TLB que forçam o acesso à tabela de páginas que está na memória. Como foram considerados três níveis de tabela de páginas, cada *miss* na TLB resulta em três acessos a memória.

O próximo teste foi feito aumentando o parâmetro de tamanho da página (*page_size*) para 1MB. Desta forma, além de aumentar o tamanho do vetor, buscou-se fazer com que os acessos a este fossem ainda mais espaçados, de modo que a requisição de endereços necessitasse novas entradas na TLB, causando assim mais *misses*. O resultado pode ser visto na figura 5.

Neste teste, a única vantagem da TLB com páginas de 4mb foi a diminuição de *misses* na ITLB, porém este número já era muito baixo para que impactasse de forma expressiva a performance do processador. Se for comparado a quantidade de acessos a memória total para os dois tamanhos de página na TLB, vemos que este é praticamente igual.

4. Conclusão

Foi verificado que a TLB com tamanho de página de 4MB se saiu muito melhor na grande maioria das vezes, tanto nos benchmarks do *SPEC2006* como no teste do *toy benchmark*. No teste com dados bastante espaçados na memória, esta teve uma performance similar à TLB com tamanho de página de 4KB. Porém devemos levar em consideração que os computadores atuais executam normalmente dezenas de processos ao mesmo tempo. Neste estudo, isto não foi levado em consideração nos testes feitos, e portanto não é possível afirmar que a TLB com páginas de 4MB são melhores que as e 4KB em todas as situações.

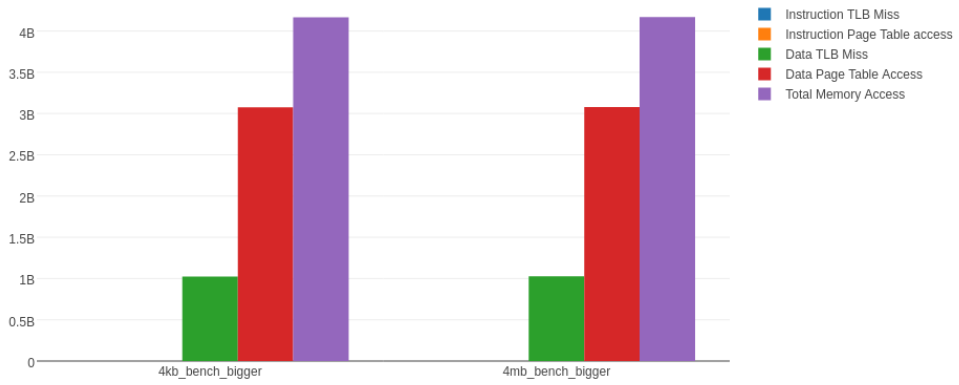

Figura 5. Comparação utilizando tabela de páginas de 4kb e 4mb para o toy benchmark utilizando um vetor maior.

Referências

Intel. Pin: Pin 3.0 user guide. In *Data de Acesso: 20 de Outubro, 2016*. https://software.intel.com/sites/landingpage/pintool/docs/76991/Pin/html/index.html.

SPEC. Spec cpu® 2006. In *Data de Acesso: 20 de Outubro, 2016*. https://www.spec.org/cpu2006/.