OProfile Estendido para Depuração de Desempenho

João Claudio Albuquerque, Roberto A Hexsel

Núcleo de Redes Sem-fio e Redes Avançadas – NR2
Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-990 – Curitiba – PR

{jclaudio,roberto}@inf.ufpr.br

Resumo. *Este artigo descreve uma extensão do programa de monitoração de desempenho* OProfile. *O* OProfile Estendido *adiciona uma referência de tempo às amostras periodicamente coletadas dos contadores de desempenho, o que permite observar o comportamento temporal das métricas. A cada amostra, o valor de todos os contadores de desempenho é registrado, o que permite relacionar o comportamento temporal de eventos com frequências distintas. Um experimento com um programa de teste que provoca faltas controladas no primeiro nível da TLB é apresentado para demonstrar a utilização do* OProfile Estendido.

1. Introdução

Contadores de Desempenho em *Hardware* (CDHs) são dispositivos de monitoramento disponíveis em CPUs de alto desempenho. Estes dispositivos possibilitam monitorar eventos ocorridos no processador durante a execução de programas e são amplamente utilizados na otimização de aplicações. Cada processador que suporta CDHs provê um conjunto de eventos observáveis, que incluem desvios tomados, faltas na cache da tabela de páginas (*Translation Lookaside Buffer*, TLB), etc. A quantidade de eventos monitoráveis e as formas de acesso aos contadores são também distintas para cada CPU.

Dentre as ferramentas disponíveis para a depuração de desempenho que utilizam CDHs existem: (i) *drivers* de acesso aos contadores [Petterson 2008]; (ii) bibliotecas para leitura e escrita dos CDHs [Berrendorf and Zeigler 2003, Heller 2001]; (iii) aplicativos de monitoramento [Anderson et al. 1997, Browne et al. 2000] que operam sem nenhuma alteração no programa estudado; e (iv) ferramentas para a visualização dos dados coletados [Rose and Reed 1999]. O *OProfile* é um instrumento de depuração de desempenho baseado na utilização dos CDHs [Levon 2003a], capaz de observar (ou monitorar) simultaneamente tantos eventos quanto permitidos pelo processador. As amostras coletadas pelo *OProfile* não contém informação referente ao instante em que um evento ocorreu, nem informações a respeito do estado dos demais contadores naquele mesmo instante.

Após a execução de um programa monitorado pelo *OProfile* obtém-se informações estatísticas a respeito dos eventos ocorridos durante a medição, tais como o número de eventos causados pelo programa observado, o número de eventos causados pelos demais binários em execução e estatísticas das medidas. O *OProfile Estendido* é uma ferramenta experimental que expande as funcionalidades do *OProfile*, ao associar a cada amostra o ciclo de relógio em que ocorreu a coleta bem como o estado dos demais contadores no mesmo instante. Essa informação complementar permite observar a evolução da contagem dos eventos ao longo do tempo. Trechos de interesse do programa monitorado

podem ser delimitados pela função `readtsc()`, que permite ler o ciclo de relógio do processador no instante da sua chamada. Essa informação pode ser registrada e utilizada no pós-processamento dos dados. Para demonstrar a capacidade do *OProfile Estendido* relatamos um experimento que investiga o comportamento da hierarquia de TLBs do processador com um programa de teste simples, escolhido por conta da facilidade em se comparar os dados medidos com os valores esperados para as métricas de interesse, que são faltas na L1d-TLB.

O restante do trabalho está organizado como segue. A Seção 2 apresenta uma breve descrição dos trabalhos relacionados enquanto a Seção 3 contém uma descrição mais detalhada do *OProfile*, bem como das alterações nele introduzidas que resultaram no *OProfile Estendido.* A Seção 4 descreve o experimento que exemplifica o uso do *OProfile Estendido*, e discute brevemente as divergências entre estimado e medido, bem como a interferência causada pelas ferramentas. A Seção 5 apresenta nossas conclusões.

2. Trabalhos Relacionados

Dentre os estudos relacionados à acurácia dos contadores dois deles [W et al. 2001, M E Maxwell 2002] comparam – para os mesmos experimentos – os valores obtidos com os contadores a valores calculados, e a valores obtidos em simulações para os mesmos experimentos. A ferramenta base utilizada no primeiro artigo é [Baer 2002] e a do segundo é [Browne et al. 2000]. Os dois trabalhos relatam e classificam divergências encontradas entre valores medidos e esperados. Em [Weaver and McKee 2008b] vários programas, características do SO e dos *benchmarks* são analisados e ajustados para reduzir o erro nas contagens em diferentes processadores da arquitetura x86. Os autores analisam apenas as contagens de instruções completadas e as alterações introduzidas aumentam a convergência dos valores das contagens entre os diferentes processadores testados. Os autores de [Weaver and McKee 2008a] afirmam que apesar dos modelos de processadores suportados pelos simuladores não comerciais serem defasados e estarem longe de simular a arquitetura dos processadores atuais, a análise com simuladores ainda se faz necessária. [Petterson 2008, Cavazos et al. 2007] apresentam ferramentas e estudos realizados, que não incluem referência de tempo nas amostras.

3. OProfile Estendido

OProfile O *OProfile* é composto por: (i) componentes dependentes de arquitetura (DA) – código responsável pela manipulação dos CDHs em cada arquitetura suportada; (ii) *oprofilefs* – pseudo sistema de arquivos que mantém algumas configurações; (iii) *CPU Buffer* e *Event Buffer* – áreas que armazenam as amostras em espaço de SO; (iv) *driver* genérico para o *kernel* – recebe as amostras enviadas pelo componente DA e as armazena no *buffer* apropriado; (v) *OProfile daemon*– responsável por receber as amostras do *driver* genérico e gravá-las em disco; (vi) ferramentas de pós-processamento – selecionam as amostras coletadas apresentando-as em formato compreensível.

Durante a execução do *OProfile* cada CDH do processador é inicializado e configurado para monitorar um determinado evento que, quando ocorre, causa um incremento do contador apropriado. No momento em que o contador atinge o valor limite, uma interrupção causa a execução do código do *driver* DA que lê o valor do PC e o número do contador em *overflow*, reiniciando-o e inserindo os valores do contador e do PC no *CPU*

Buffer. O código do *driver* DA também registra no *CPU Buffer* as trocas de processos em execução, assim como as trocas de modo de execução entre usuário e sistema.

Periodicamente o *CPU Buffer* é sincronizado com o *Event Buffer* pelo *driver* genérico do *OProfile* que, além dos dados das amostras e do PC, insere no *Event Buffer* um identificador do binário em execução no momento da leitura da amostra. Os dados são então transferidos para o espaço de usuário pelo *OProfile daemon*, que trata estes dados mapeando em disco arquivos de amostras para os programas em execução. Para cada binário é mantido um conjunto de arquivos, um arquivo para cada evento monitorado. O arquivo contém registros com o valor do PC no momento em que a interrupção do contador foi atendida, além do número de interrupções já atendidas para cada um dos valores de PC registrados. Uma descrição detalhada encontra-se em [Levon 2003b].

Extensões O tratador da interrupção do *driver* DA foi alterado para ler, além do valor do PC e do contador em *overflow*, os valores dos demais CDHs e também a referência de tempo do *Time Stamp Counter* (TSC), que é um contador de 64 bits que mantém o número de ciclos de relógio do processador desde a sua inicialização. As estruturas *CPU Buffer* e *Event Buffer* também foram alteradas para armazenar estes novos dados, assim como o *driver* genérico, que realiza a sincronização entre os dois *buffers*.

O *OProfile daemon* foi alterado para mapear dois novos arquivos, além dos arquivos normais: o primeiro contém um índice que relaciona nomes de binários e *hashes*, e o segundo mantém, para cada ciclo do TSC lido, os valores de todos os CDHs além de informações complementares como o *hash* do nome do binário em execução naquele instante. Esses registros de tempo absoluto (do ponto de vista do processador) atrelados aos valores dos contadores permitem gerar gráficos que demonstram o comportamento dos eventos monitorados ao longo do tempo, ao invés do simples 'histograma' de eventos fornecido pelo *OProfile.*

4. Operação da TLB

Nesta seção descrevemos o experimento projetado para validar o *OProfile Estendido.* O programa de teste aloca e percorre 500 vezes um conjunto de n páginas. Em seguida, um novo conjunto de tamanho $n + 1$ páginas é alocado em uma região distinta da memória – para evitar a reutilização dos mapeamentos das TLBs – e esta nova área é também percorrida 500 vezes. O número inicial e final de páginas percorridas são definidos pelo usuário. Espera-se que o programa de teste provoque um número de faltas na TLB condizente com o de estimativas que são facilmente calculadas.

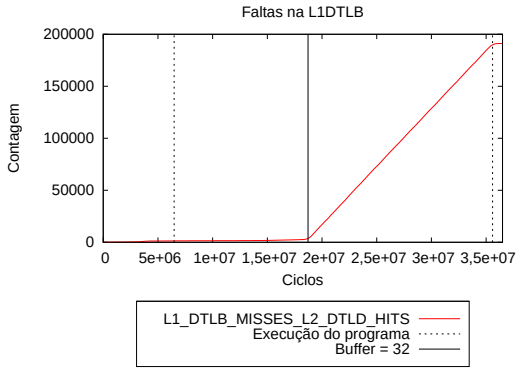
Os resultados apresentados a seguir são a média de 20 execuções do teste. Nos dois gráficos mostrados na Figura 1, as linhas verticais pontilhadas representam o início e o fim da execução do programa de teste, a linha vertical sólida representa o limite do primeiro nível da TLB durante o intervalo de páginas percorrido, o eixo vertical é o número de ocorrências do evento monitorado ao longo do tempo, que é o eixo horizontal.

Faltas na L1d-TLB A hierarquia de TLBs do AMD Athlon XP possui dois níveis. A TLB de primeiro nível, L1d-TLB contém 32 mapeamentos, é totalmente associativa com reposição LRU (*least recently used*) [Inc 2004]. É esperado que, antes da utilização de 32 páginas, cada ciclo de 500 acessos acumule apenas o número de faltas correspondente ao espaço alocado: um *buffer* de 27 páginas causa 27 faltas na primeira volta do *loop*,

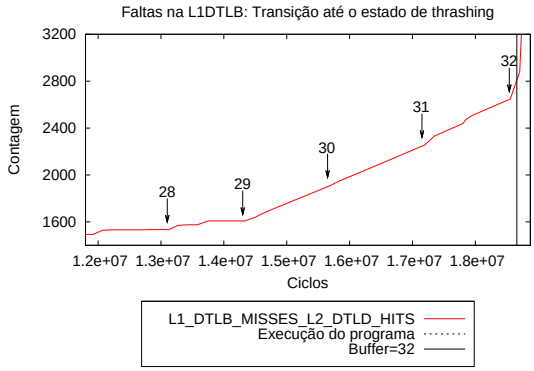
enquanto que nas outras 499 voltas nenhuma falta ocorre porque os 27 mapeamentos já estão carregadas na TLB. Acima do limite de 32 páginas, espera-se que todos os acessos realizados em todas as iterações do *loop* causem faltas, o que caracteriza o estado de *thrashing* na L1d-TLB. Isso é esperado porque, sendo a reposição por LRU, cada nova página mapeada além das 32 expurga um mapeamento que será reutilizado em breve.

Considerando que programa usa ao menos uma página adicional para sua pilha, pode-se considerar que o limite de *thrashing* na L1d-TLB é 31 páginas. Desta forma, o número de faltas esperado quando o programa de testes percorre de 22 a 31 faltas é o somatório de todos estes valores: 265 faltas. Na segunda metade do experimento, quando são percorridas de 32 a 41 páginas, o valor estimado é a soma dos valores do intervalo multiplicado pelo número de iterações do *loop*: 182.500 faltas. O total de faltas esperado é 182.765 faltas, que é a soma dos valores estimados nas duas etapas do teste.

A Figura 1(a) mostra a contagem de faltas na medida em que o tamanho do bloco percorrido aumenta de 22 a 41 páginas. Antes da medição atingir o ponto de interesse, após a carga do programa, faltas decorrentes da alocação da pilha e de dados do programa são registradas, totalizando 1.258 faltas. No momento em que o teste propriamente dito se inicia, as faltas crescem linearmente com o tamanho da área alocada pois ainda há registros livres na L1-TLB. Quando são alocadas mais de 28–30 páginas, a L1d-TLB entra em *thrashing*, como esperado. Ao final do teste, desprezados os valores anteriores à execução do programa, são contabilizadas 188.953 faltas, 3% acima do esperado.



(a) Faltas na L1d-TLB durante a execução.



(b) Evolução das faltas na L1d-TLB até o *thrashing*

Figura 1. Faltas na L1d-TLB.

É importante observar que a estimativa para o número de faltas não modela fielmente o que ocorre no processador, enquanto este executa o programa sobre um SO multitarefa, concorrentemente às demais aplicações. A deficiência da estimativa torna-se evidente quando se observa a ampliação do gráfico na Figura 1(b): cada flecha indica o tamanho do *buffer* a ser percorrido (número de páginas). É possível observar que entre o regime normal e o de *thrashing*, há um regime intermediário – entre 29 e 31 páginas – no qual a taxa de faltas aumenta muito, possivelmente em virtude dos conflitos de mapeamento entre o programa de teste e os demais processos, mas ainda não o suficiente para caracterizar o *thrashing*, evidenciado a partir do ponto em que o *buffer* atinge o tamanho de 32 páginas, quando a inclinação do gráfico se aproxima da vertical.

É possível observar degraus no gráfico da Figura 1(b), o que nos permite con-

tar o número de faltas para cada tamanho de *buffer* percorrido. A Tabela 1 apresenta a comparação entre valores contados nos degraus e os valores estimados para alguns trechos da execução do programa, e o total de faltas medidas no experimento.

Tabela 1. Faltas na L1d-TLB.

INTERVALO [pág]	MEDIDO	ESTIMADO
23	42	23
24	33	24
36	18.427	18.000
37	20.197	18.500
Total	188.953	182.765

Interferência e divergência nas medidas A diferença entre o número de ciclos, com e sem a medição, é utilizada para medir a interferência causada pelas ferramentas de monitoramento. A Tabela 2 mostra as contagens de ciclos registradas pelo TSC durante a execução do programa de teste, obtidas de três maneiras: (i) com o *OProfile Estendido*; (ii) com o *OProfile* original; e (iii) sem o *OProfile*. A medida de tempo nos dois últimos é obtida com duas invocações da função `readtsc()`. A interferência causada pelas medidas com as duas versões do *OProfile* é da ordem de $2,1\%$.

Tabela 2. Contagem de ciclos do programa de teste, $\times 10^6$.

INTERVALO	OP EXT		OP ORIG		—
22 a 41 páginas	29,1	(2,1%)	29,1	(2,1%)	28,5

A divergência entre as medidas de número de faltas, obtidas com o *OProfile* e o *OProfile Estendido* é mostrada na Tabela 3. A diferença é devida ao método de leitura dos contadores em cada uma das ferramentas: enquanto o *OProfile* não conta 'faltas', mas sim "o número de *overflows* ocorridos", o número (aproximado) de eventos é o produto do número de interrupções pelo valor de configuração do contador (35.000 faltas neste experimento). No *OProfile Estendido*, o processador é interrompido pelo contador de ciclos de execução (`Clock_Unhalted`) a cada 150.000 ciclos, e os valores nos contadores de faltas são então registrados e atualizados a cada interrupção.

Tabela 3. Diferença na contagem de faltas na L1d-TLB, $\times 10^3$.

INTERVALO	OP EXT	OP ORIG	DIFERENÇA
22 a 41 Páginas	188	137	37%

5. Conclusão

Apresentamos uma extensão do *OProfile* que permite gerar perfis de execução que associam as métricas disponíveis no processador com informação de tempo absoluto. Com o *OProfile Estendido* é possível tomar amostras temporais de todos os contadores de eventos observados a uma taxa de amostragem tão alta quanto permitida pelo projeto da CPU.

Relatamos um experimento que demonstra a utilização do *OProfile Estendido*, no qual um programa de teste provoca faltas controladas no primeiro nível da TLB de dados, de forma que seja possível observar a evolução das faltas nesta estrutura na medida que

o conjunto de trabalho aumenta. As faltas são monitoradas até que seja atingido o estado de *thrashing* na L1-TLB. Os resultados indicam que a medida é confiável, e permitem observar o estado transitório antes do *thrashing* na L1d-TLB. Também foi constatado que para a configuração do teste utilizada neste trabalho, o *OProfile Estendido* não causa interferência maior do que o *OProfile* original.

Referências

[Anderson et al. 1997] Anderson, J. M. et al. (1997). Continuous profiling: where have all the cycles gone. *ACM Trans on Computer Systems*, 15(4):357–390.

[Baer 2002] Baer, T. (2002). lperfex: A hardware performance monitor for Linux/IA32 Systems. `http://www.osc.edu/~troy/lperfex/`.

[Berrendorf and Zeigler 2003] Berrendorf, R. and Zeigler, H. (2003). PCL, the Performance Counter Library, version 2.3. `http://www.fz-juelich.de/jsc/PCL/`.

[Browne et al. 2000] Browne, S., Dongarra, J., Garner, N., Ho, G., and Mucci, P. (2000). A portable programming interface for performance evaluation on modern processors. *The Intl Journal of High Performance Computing Applications*, 14(3):189–204.

[Cavazos et al. 2007] Cavazos, J., Fursin, G., Agakov, F., Bonilla, E., O'Boyle, M. F. P., and Temam, O. (2007). Rapidly selecting good compiler optimizations using performance counters. In *IEEE/ACM Intl Symp on Code Generation and Optimization*, pages 185–197, Los Alamitos, CA, USA. IEEE Computer Society.

[Heller 2001] Heller, D. (2001). Rabbit: A performance counters library for Intel/AMD processors and Linux. `http://www.scl.ameslab.gov/Projects/Rabbit`.

[Inc 2004] Inc, A. M. D. (2004). AMD Athlon processor x86 code optimization guide.

[Levon 2003a] Levon, J. (2003a). OProfile - A System Profiler for Linux. `http://oprofile.sourceforge.net/news/`.

[Levon 2003b] Levon, J. (2003b). OProfile Manual. `http://oprofile.sourceforge.net/docs/index.php3/`.

[M E Maxwell 2002] M E Maxwell, P. J. T. e. L. S. (2002). Accuracy of performance monitoring hardware. *Proc LACSI Symposium*.

[Petterson 2008] Petterson, M. (2008). Linux x86 performance-monitoring counter's driver, ver. 2.6.39. `http://user.it.uu.se/~mikpe/linux/perfctr`.

[Rose and Reed 1999] Rose, L. D. and Reed, D. A. (1999). SvPablo: A multi-language performance analisys system. In *Proc 1999 Intl Conf on Parallel Processing*, pages 311–318.

[W et al. 2001] W, W. K., Teller, P. J., and Castillo, G. (2001). Just how accurate are performance counters? *IEEE Performance, Computing, and Communications*, pages 303–310.

[Weaver and McKee 2008a] Weaver, V. M. and McKee, S. A. (2008a). Are cycle accurate simulations a waste of time? In *6th Workshop on Duplicating, Deconstructing, and Debunking*.

[Weaver and McKee 2008b] Weaver, V. M. and McKee, S. A. (2008b). Can hardware performance counters be trusted? In *IISWC*, pages 141–150.