Uso de Cache de Pilha em Processadores Atuais

Eduardo Guerra[1], **Francis B. Moreira**[1], **Phillipe O. A. Navaux**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{eguerra,fbmoreira,navaux}@inf.ufrgs.br

Resumo. *Em processadores de uso geral, é comum as aplicações utilizarem uma área de memória designada a uma pilha de execução, a fim de manter informação sobre o estado de subrotinas e armazenar variáveis locais a estas subrotinas. Dada a frequência de uso da pilha, otimizar os acessos a esta região de memória pode resultar em ganhos de desempenho na execução de aplicações. Sendo assim, este artigo tem como objetivo explorar o uso de uma memória cache dedicada exclusivamente à área de pilha na memória e analisar os impactos causados por ela no desempenho do processador.*

1. Introdução

A área de pilha é uma porção de memória contígua alocada por cada thread em execução, a qual mantém o contexto do escopo atual em que a aplicação se encontra. A sua alocação é feita no início da aplicação e pode crescer durante a execução, essa porção de memória empilha os endereços de retorno de subrotinas, assim como variáveis locais durante a execução de blocos de código. Neste artigo, é proposto o uso de uma memória cache adicional, a qual armazenaria apenas valores presentes na região da memória da pilha.

Além de remover um fluxo de dados da cache de dados, que não precisaria mais guardar os valores vindos da região da pilha, a nova memória cache pode ser acessada mais eficientemente, ao escolher 1 projeto focado em tempo de acesso. Assim, existe a possibilidade de aumentar a taxa de acertos na cache de dados e reduzir a latência para os acessos feitos na área da pilha. Espera-se portanto que o uso de uma cache de pilha resulte em um ganho de desempenho no processador.

2. Motivação

Para estudar a taxa de utilização da pilha, foi utilizado o Pin [Luk et al. 2005], um instrumentador dinâmico de binários da Intel. Através deste, foi possível contar o número total de acessos à memória realizados pela aplicação instrumentada, assim como o número de acessos à área de pilha na memória.

Como é possível ver na Tabela 1, o número médio de acessos à pilha é de 23% do número total de acessos para a carga de benchmarks paralelos da NASA (NPB) de classe A executados por 8 threads. Portanto, espera-se que reduzir o tempo de acessos à área de pilha leve a ganhos de desempenho no processamento, já que estes acessos constituem parte considerável dos acessos à memória. Atualmente, processadores convencionais comumente possuem uma latência de acesso à cache de dados de 4 ciclos. Porém, a latência de acesso à uma memória cache pode ser reduzida ao simplificar sua configuração.

Normalmente, a área de pilha é pequena em relação à aplicação, e possui alta localidade espacial e temporal [Alshegaifi and Huang 2016]. Assim, é possível reduzir a

latência de acesso ao utilizar uma cache menor e mais simples. Na Tabela 2 são demonstrados tempos de ciclo de caches do simulador FinCACTI [Shafaei et al. 2014] baseados em uma tecnologia de 7nm, para um processador de frequência 3.33 Ghz (0.3 ns de ciclo). Aqui, mostra-se que o mapeamento direto influi fortemente no tempo de acesso à cache. Este mapeamento normalmente gera conflitos entre endereços, resultando em subutilização do espaço da cache e uma taxa de erros mais alta. Porém, assumindo que a localidade da pilha em um escopo limita seu uso a regiões pequenas, essa desvantagem não existe.

Table 1. Porcentagem de acessos à região de memória da pilha.

Benchmark	Acessos à Pilha	Acessos Normais	Total	Porcentagem
bt.A.x	55196892056	98930861522	154127753578	36
cg.A.x	36949882	3191073910	3228023792	1
ep.A.x	6101896235	6656570233	12758466468	48
ft.A.x	396415853	8881748247	9278164100	4
is.A.x	75797435	1520603248	1596400683	5
lu.A.x	4380719119	16070986652	20451705771	21
mg.A.x	1318181417	5172286702	6490468119	20
sp.A.x	5836812962	81979881482	87816694444	7
ua.A.x	16571936500	85406001858	101977938358	16
Média	9990622384	34201112650	44191735035	23

Como discutido anteriormente, o tamanho da área de pilha possui localidade espacial entre escopos, e localidade temporal para um escopo. Assim, pode-se utilizar o mapeamento direto sem sofrer perdas significativas de desempenho causados por conflitos de linhas. Para os testes realizados na Seção de testes, o tamanho escolhido será 16KB, tornando possível o armazenamento simultâneo de múltiplas páginas de memória da pilha, enquanto mantém-se o tempo de acesso baixo.

Table 2. Tempos de acesso, em ciclos de processador, para parâmetros de cache diferentes.

Tamanho da Cache	Associatividade	Tempo de Ciclo	Ciclos de Processador
4 KB	1 via	0.221225 ns	1 ciclos
4 KB	2 vias	0.721745 ns	2 ciclos
4 KB	4 vias	2.64095 ns	9 ciclos
8 KB	1 via	0.221225 ns	1 ciclos
8 KB	2 vias	0.721745 ns	2 ciclos
8 KB	4 vias	2.64095 ns	9 ciclos
16 KB	1 via	0.221225 ns	1 ciclos
16 KB	2 vias	0.721745 ns	2 ciclos
16 KB	4 vias	0.721745 ns	2 ciclos
32 KB	8 vias	0.721745 ns	2 ciclos

3. Metodologia

Para realização dos testes de uso da região da pilha, foi utilizado o instrumentador Pin [Luk et al. 2005], versão 2.14. Para os experimentos de desempenho, foi utilizado

o simulador ZSim [Sanchez and Kozyrakis 2013]. Para as simulações, o código fonte do simulador foi modificado para detectar os acessos à área de pilha e direcioná-los para uma cache de pilha. A detecção é feita utilizando o instrumentador Pin-[Luk et al. 2005], que verifica se os acessos à memória foram realizados pelos registradores dedicados à pilha, *Stack Pointer* e *Base Pointer*. A configuração da máquina simulada possui 8 cores com execução fora de ordem, baseados na arquitetura Westmere [Kurd et al. 2010], com frequência de 3.3 Ghz. A configuração das caches está detalhada na Tabela 3.

Table 3. Parâmetros das memórias cache do sistema simulado.

Cache	L1 Dados	L1 Pilha	L1 Instruções	L2	L3
Tamanho	32KB	16KB	32KB	256KB	8MB
Associatividade	8 vias	1 via	8 vias	4 vias	16 vias
Latência	4 ciclos	1 ciclo	4 ciclos	14 ciclos	34 ciclos

O conjunto de benchmarks usado foi o NAS Parallel Benchmarks (NPB) [Bailey et al. 1991]. Todas aplicações foram executadas, com exceção à aplicação DC. A classe "A" foi usada para todos os benchmarks, por tratar-se da menor classe inclusa na lista de classes de problemas de testes padrão, reduzindo o tempo de simulação. Os benchmarks foram compilados com gcc 5.4.0.0.

4. Experimentos

Nesta sessão, é feita uma análise sobre o desempenho de um processador com cache para pilha. Os resultados referentes ao benchmark NPB são mostrados na Figura 1, expressos em números de ciclos de relógio utilizados pela thread principal de cada simulação. Na primeira coluna, encontra-se o número de ciclos utilizado pela aplicação sem a presença da cache de pilha. Na segunda coluna, encontra-se o número de ciclos utilizados pela mesma aplicação com a atuação da cache de pilha.

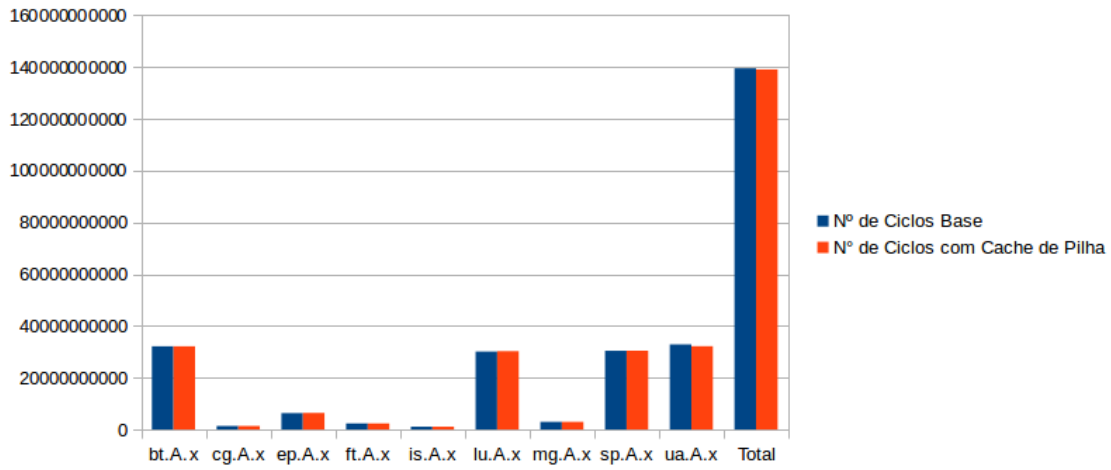

Figure 1. Tempo de execução em ciclos de processador para benchmarks NPB com 8 threads.

Como é possível observar na Figura 1, os resultados obtidos não condizem com a expectativa. A presença da nova memória cache teve impacto negligível. Aqui, é importante notar peculiaridades na implementação de coerência de cache do simulador, o qual

apresentou piora no desempenho em testes com latência de invalidação reduzida. Apesar da cache de pilha ser mais rápida, ela possui uma taxa de erros muito alta, o que seria justificável caso ocorra baixa localidade na área de pilha. Assim, temos evidência contrária aos resultados obtidos em [Alshegaifi and Huang 2016].

5. Conclusão

Os testes finais não demonstraram diferença significativa na execução dos benchmarks ao utilizar uma cache exclusiva para a área de pilha. Porém, existem indicativos de que a localidade espacial da cache de pilha precisa ser melhor explorada. No trabalho futuro será desenvolvido um *prefetcher* para utilizar a localidade espacial da cache de pilha. Deve-se também explorar o funcionamento da coerência de cache e analisar a implementação desta no simulador ZSim, o qual não parece suportar tal modificação.

References

Alshegaifi, A. K. and Huang, C.-H. (2016). Impact of stack caches: Locality awareness and cost effectiveness. *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, 10(4):545–551.

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., et al. (1991). The nas parallel benchmarks. *The International Journal of Supercomputing Applications*, 5:63–73.

Kurd, N. A., Bhamidipati, S., Mozak, C., Miller, J. L., Wilson, T. M., Nemani, M., and Chowdhury, M. (2010). Westmere: A family of 32nm ia processors. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 96–97. IEEE.

Luk, C.-K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V. J., and Hazelwood, K. (2005). Pin: building customized program analysis tools with dynamic instrumentation. In *Acm sigplan notices*, volume 40, pages 190–200. ACM.

Sanchez, D. and Kozyrakis, C. (2013). Zsim: fast and accurate microarchitectural simulation of thousand-core systems. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 475–486. ACM.

Shafaei, A., Wang, Y., Lin, X., and Pedram, M. (2014). Fincacti: Architectural analysis and modeling of caches with deeply-scaled finfet devices. In *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, pages 290–295. IEEE.