Avaliação de Algoritmos de Previsão de Desvios

Guilherme Zamberlam Pomini[1]

[1]Departamento de Informática (DIN) – Universidade Estadual de Maringá (UEM)
Sede Maringá – Paraná – Brasil

guizamberlam@hotmail.com

Abstract. *This paper presents the concept of Branch Prediction Algorithms (BPD) to optmize the processing of a Pipeline. Due to the existence of numerous BPDs with different approaches an evaluation is necessary to define the best BPD in a given situation. The article exposes experiments performed with four different branch predictors (2bits, Bimode, Tournament and Ltage) and the evaluation of their performance. Through the experiments it was possible to conclude that the most cost-effective predictor of those proposed in this article was the Ltage.*

Resumo. *Este artigo apresenta o conceito de Algoritmos de Previsão de Desvios (APD) para otimizar o processamento de um Pipeline. Devido a existência de inúmeros APDs com abordagens diferentes se faz necessário uma avaliação para definir o melhor APD para determinada situação. O artigo expõe experimentos realizados com quatro diferentes preditores de desvio (2bits, Bimode, Tournament e Ltage) e a avaliação de desempenho dos mesmos. Através dos experimentos foi possível concluir que o preditor com maior custo-benefício dos propostos nesse artigo é o Ltage.*

1. Introdução

Dentro do âmbito da Arquitetura de Computadores existe uma tecnica aplicada no Hardware chamada Pipeline que busca otimizar o fluxo de execução de instruções no processador. A ideia central do *Pipeline* é dividir o processamento de uma instrução em etapas, com essa divisão é possível ter várias instruções em diferentes estados de execução ao mesmo tempo, otimizando a capacidade do processador [Dal Pizzol 2005]. Contudo, a abordagem pipeline sofre com o processamento de instruções de desvio, sendo necessário parar a entrada de instuções no *Pipeline* até o término do processamento do desvio para que se saiba de fato a próxima instrução a ser executada, assim deixando a execução mais lenta e não aproveitando todo o potencial do processador [Patterson and Hennessy 1990].

Uma abordagem para eliminar a necessidade de esperar a instrução de desvio terminar sua execução é adivinhar se o mesmo será ou não tomado. Tal método de adivinhação permite a entrada das próimas instruções no pipeline continuar a fluir mesmo quando um desvio ainda não terminou sua execução, tendo como empecilho somente se o "chute"foi incorreto. Para aumentar a acurácia do "chute"tomado pelo processador na hora de analisar um desvio foram criados os Algoritmos de Previsão de Desvio (APD).

Tendo em vista a existência de diversos tipos de APDs com suas diferentes abordagens, o objetivo desse artigo é apresentar a análise de desempenho de quatro diferentes

APDs, sendo eles: 2bits, Bimode, Ltage e Tournament. Para a realização da análise foram feitas simulações utilizando cinco diferentes programas, com cada um recebendo os quatro APDs, totalizando vinte simulações.

Este artigo foi organziado em cinco seções. A seção 2 introduz o simulador Gem5 utilizado para a realização dos testes. A seção 3 apresenta o de um APD e a abordagem de cada um utilizado neste artigo. A seção 4 mostra a metodologia utilizada e os resultados analisados. Por fim a seção 5 apresenta a conclusão do artigo a partir dos resultados.

2. Simulador Gem5

O Gem5 é uma infraestrutura de simulação de sistemas de computadores que une os simuladores M5 e GEMS [Binkert et al. 2011]. O Gem5 tem como objetivo criar um ambiente altamente customizável e flexível para a simulação de diversos tipos de CPUs proprias do Gem5 (AtomicSimple, TimingSimple, InOrder e O3) e arquiteturas reais como ARM, ALPHA, MIPS, Power, SPARC, e x86. O simulador possui dois modos de execução, sendo eles:

- **Full System Emulation** - executa um modelo de sistema computacional completo incluindo um sistema operacional e controles de entrada e saída.
- **System Call Emulation** - elimina a necessidade de um sistema operacional e outros mecanismos emulando a maioria dos serviços do sistema.

3. Algoritmos de Previsão de Desvio

Como forma de aumentar o desempenho do *Pipeline* foram criados mecanismos para prever o resultado de uma instrução de desvio durante o processamento, tal abordagem reduz o atraso no *Pipeline* mas ainda sofre penalidades quando a previsão é incorreta [Smith 1981]. Para aumentar a taxa de acerto do "chute"para cada desvio foram criados Algoritmos de Previsão de Desvio, um conjunto de técnicas tanto de Software quanto de Hardware que servem como auxílio nas tomadas de decisões em um fluxo de execução.

Os preditores de desvio existem nas mais variadas formas, com diferentes comportamentos e abordagens, desde predições estáticas como todo salto salto para frente no fluxo ser tomado e todo salto para trás não ser tomado, quanto predições dinâmicas analisando tabelas de histórico de cada salto. Dito isso, nessa seção será apresentado os quatro preditores analisados.

Ltage O algoritmo LTage é uma combinação do preditor Tage com um preditor de loop. O componente de loop tenta identificar loops regulares com um um numero constante de iterações. O componente Tage é derivado de outros preditores, possui um preditor base que gera uma predição de partida, tal predição é tratada por outros preditores rotulados com diferentes tamanhos de históricos formando uma série geométrica, tendo como resultado uma predição que combina com o maior histórico [Seznec 2007]. A Figura 1 ilusta o processamento do preditor Ltage com cinco componentes.

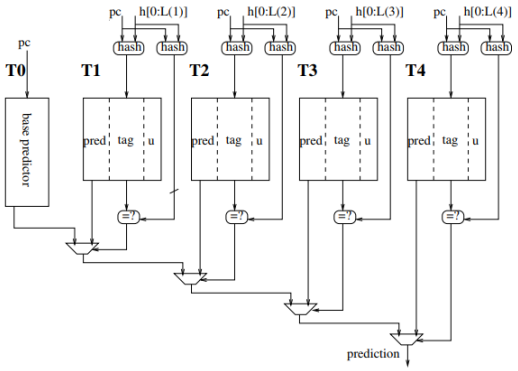


Figura 1. Estrutura Ltage

Bimode O preditor Bimode é um algoritmo de dois níveis que trabalha com um historico de previsão global e o endereço do desvio que indexam tabelas com contadores de dois bits [Lee et al. 1997]. Conforme o padrão do histórico global um dos preditores de direção é ativado para depois o preditor de escolha dar uma previsão final para o salto, como ilustra a Figura 2.

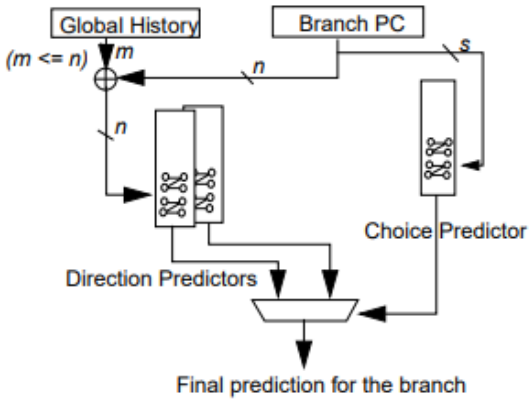


Figura 2. Estrutura Bimode

LocalBP (2bits) LocalBP vem de *Local Branch Predictor*, um preditor de desvio que guarda um histórico local de cada desvio [Patterson and Hennessy 1990]. O algoritmo se baseia na ideia de prever um padrão para os desvios baseado no seu historico utilizando contadores saturados de 2 bits. Cada salto possui seu contador que é alterado a cada ciclo, sendo incrementado se o salto é tomado e decrementado caso contrário como mostra a Figura 3

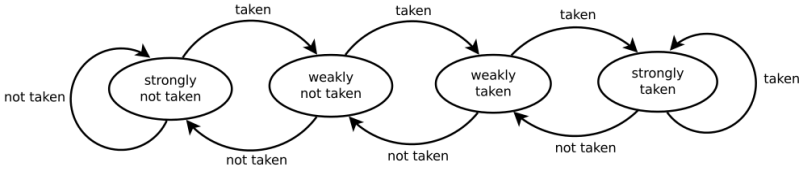


Figura 3. Contador do LocalBP

Tournament O algoritmo Tournament é um híbrido de outros previsores, por si só funciona como um seletor realizando uma espécie de “torneio”para decidir qual a

melhor abordagem para cada salto [Patterson and Hennessy 1990]. A seleção se da entre um preditor local que possui uma tabela de de histórico de saltos para cada salto, e entre um preditor global que possui uma tabela de historico global de todos os saltos processados como ilustra a Figura 4. O algoritmo Tournament trabalha em cima de que existem saltos que são dependentes de informações locais enquanto outros são influenciados por informações globais.

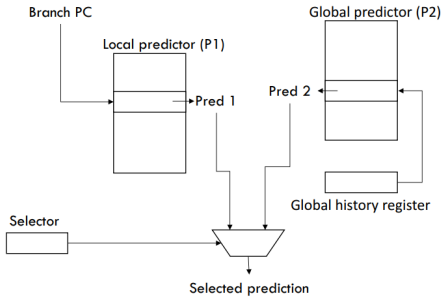


Figura 4. Estrutura Tournament

4. Avaliação

Nesta seção será abordado a metodologia utilizada na avaliação e os resultados oriundos da mesma.

4.1. Metodologia

Para a avaliação dos Algoritmos de Previsão de Desvio foram feitas simulações usando o simulador Gem5.

Software Os Softwares utilizados nos experimentos foram o simulador Gem5, e os compiladores GCC e G++. Todos executados em um sistema Ubuntu 16.04.4 LTS x64.

Hardware Os testes com o gem5 foram realizado em uma máquina em nuvem do servidor Google Cloud com as especificações: Processador Intel Xeon E5-2699, Memória RAM de 16GB e Sistema Operacional Ubuntu 16.04.4 LTS x64.

Programas Utilizados Os preditores de desvio agiram em cima de quatro programas de entrada com diferentes propósitos como:

1. **Gramschmidt** Solutor de Álgebra Linear
2. **h264dec** Decodificador de arquivos de vídeo
3. **Oopack_v1p8** Comparador de desempenho entre C e C++
4. **mandel-2** Exibe uma figura formada por astericos

Para a compilação de cada programa foram utilizadas as flags -O3 para um maior nível de otimização e -static para eliminar dependências de bibliotecas dinâmicas. Somente o programa Oopack_v1p8 foi compilado com o compilador g++, para os demais foi utiizado o gcc.

Gem5 A simulação no Gem5 foi feita em modo SE (System Call Emulation), não necessitando um sistema operacional. Com o gem5 foi construído o executável do simulador de arquitetura X86 utilizando a abordagem “fast” que otimiza a arquitetura ao máximo de perfomance e ignora possíveis checagens de erro.

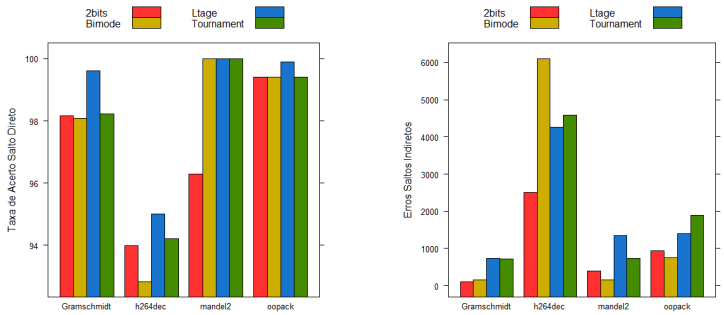
Arquitetura Simulada Foi modelado uma arquitetura de base para o simulador utilizando a linguagem Python com as especificações: Arquitetura X86, 2GHz de Clock, 1 Processador, CPU O3, Cache L1 separada, Cache L2 unificada e Memória DDR3_1600_8x8. A arquitetura foi parametrizada para receber um programa executável (binário) como entrada e um preditor de desvio.

Experimentos Cada um dos programas foi testado com os quatro preditores de desvios no gem5, totalizando 16 simulações. Após cada simulação foi armazenado o arquivo stats.txt com os dados da mesma como a quantidade de tentativas de predição, quantidade de predições incorretas, entre outros. Tais arquivos stats serviram como base para a avaliação de cada preditor.

Avaliação O artigo visa analisar o melhor preditor de saltos diretos, de saltos indiretos, o que faz o melhor uso do endereço de retorno (RAS) e as diferenças no tempo de execução.

4.2. Resultados

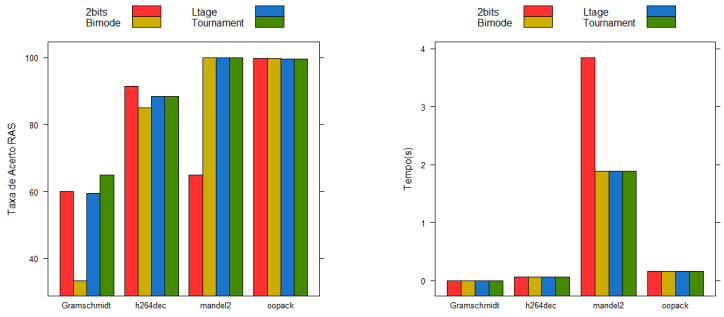
Nesta seção será apresentado os resultados dos testes de cada programa de entrada com base nos gráficos de acerto dos desvios (Fig. 5(a)), acerto da RAS (Fig. 6(a)), Tempo de Execução (Fig. 6(b)) e Erro de Previsão de Saltos Indiretos (Fig 5(b)).



(a) Taxa de Acerto Previsão de Saltos Condicionais

(b) Erro de Previsão de Saltos Indiretos

Figura 5. Previsão de Saltos



(a) Taxa Acerto RAS

(b) Tempo de Execução

Figura 6. Previsão RAS e Tempo de Execução

4.2.1. Previsão de Saltos Condicionais

A taxa de acerto na previsão de saltos condidicionais teve valores proximos de 100% para os quatro preditores, com o menor valor encontrado sendo de 92% com o preditor Bimode no programa h264dec como ilustra a Fig. 5(a). Dos quatro preditores o que se destaca nessa previsão é o Ltage em que nos quatro experimentos possuiu a maior taxa de acerto ou empatou com os demais como no caso do mandel-2. Seguindo o Ltage vem o Tournament que se sobressaiu ao 2bits e Bimode em metade dos casos. A análise do pior preditor de desvio é de certa forma sutil, com os preditores Bimode e 2bits tendo o pior desempenho em um experimento cada e uma taxa semelhante nos demais.

4.2.2. Previsão de Saltos Indiretos

Nesta previsão é possível notar um comportamento semelhante entre os preditores Ltage e Tournament que em 75% dos casos tiveram a quantidade de erros indiretos maior que a do Bimode e 2bits. O único caso em que o Ltage e o Tournament não foram os mais custosos foi com o programa h264dec onde o Bimode os supera por uma margem considerável como mostra a Figura 5(b). Vale notar que o comportamento do preditor Bimode é o melhor no geral, onde somente em um caso acabou se tornando o mais custoso.

4.2.3. Acerto RAS

Na análise do uso do endereço de retorno na previsão foi possível notar um comportamento similar novamente entre o Ltage e o Tournament, com a mesma taxa em três testes e tendo os maiores acertos na média. Analisando a Figura 6(a) pode-se afirmar que o destaque negativo vai para o Bimode que teve um desempenho inferior em metade dos casos, com um acerto menor que a metade dos outros três preditores no experimento do programa Gramschmidt.

4.2.4. Tempo de Execução

O tempo de execução de cada preditor se mostrou similar com três dos quatro programas testados onde a diferença entre cada preditor não era significativa, ficando na casa dos milésimos de segundos. Como mostra a Figura 6(b) somente nos testes do programa mandel-2 o comportamento foi diferenciado, com o preditor 2bits tomando um tempo de execução duas vezes maior que os demais. Tal comportamento atípico torna difícil a afirmação a análise do tempo de execução, é possível afirmar que na média os quatro algoritmos se comportam com o mesmo tempo e que o 2bits possui um desvio maior.

5. Conclusão

Neste artigo foi realizada a análise de quatro Algoritmos de Previsão de Desvio, cada um com uma abordagem diferente com o objetivo de melhorar o processamento do Pipeline. Foram realizadas simulações de desempenho com o auxílio do simulador Gem5 e cinco programas diferentes de entrada. A partir das simulações foi possível montar gráficos e analisar objetivamente o desempenho de cada preditor de desvio.

A partir dos dados analisados foi possível concluir que o melhor Algoritmo de Previsão de Desvio dos analisados foi o Ltage, possuindo uma taxa de erro nitidamente menor na previsão de saltos condicionais em todos os testes. Em segundo lugar vem o Tournament que possui que embora possua um comportamento quase similar ao 2bits e o Bimode, ainda consegue se sobressair nos testes.

O contrário pode ser notado na predição de saltos indiretos onde o Ltage possui um comportamento similar ao do Tournament como os dois que possuem a maior taxa de erro, nesse quesito os destaques vão para o Bimode e 2bits, com o último possuindo o melhor desempenho.

Ainda assim, a diferença de performance do algoritmo Ltage na previsão dos desvios é significativa em todos os testes, sendo possível afirmá-lo como o Algoritmo de Previsão de Desvio de melhor desempenho dos analisados neste artigo.

Referências

Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., et al. (2011). The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7.

Dal Pizzol, G. (2005). Previsão de desvios em arquiteturas multitarefas simultâneas.

Lee, C.-C., Chen, I.-C. K., and Mudge, T. N. (1997). The bi-mode branch predictor. In *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, pages 4–13. IEEE Computer Society.

Patterson, D. A. and Hennessy, J. L. (1990). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Seznec, A. (2007). A 256 kbits l-tage branch predictor. *Journal of Instruction-Level Parallelism (JILP) Special Issue: The Second Championship Branch Prediction Competition (CBP-2)*, 9:1–6.

Smith, J. E. (1981). A study of branch prediction strategies. In *Proceedings of the 8th Annual Symposium on Computer Architecture*, ISCA '81, pages 135–148, Los Alamitos, CA, USA. IEEE Computer Society Press.