MO601 – Arquitetura de Computadores II – Projeto 3

Uglaybe Piell Fernandes[1]

[1]Faculdade de Engenharia Mecânica – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

uglaybe@gmail.com

Abstract. *This paper aims get more data about the performance of the branch predictor proposed in the paper "Wormhole: Wisely Predicting Multidimensional Branches" presented during MICRO-47 by Jorge Albericio, Joshua San Miguel, Natalie Enright Jerger, and Andreas Moshovos comparing the performance with the same predictor, but different parameters.*

Resumo. *Este artigo busca levantar mais dados sobre a performance do branch predictor proposto no paper "Wormhole: Wisely Predicting Multidimensional Branches" apresentado durante a MICRO-47 por Jorge Albericio, Joshua San Miguel, Natalie Enright Jerger, e Andreas Moshovos, comparando a performance com o mesmo preditor com diferentes parâmetros.*

1. Introdução

Branches predictors representam um grande campo para melhoria em microarquitetura de computadores devido ao fato que *mispredictions* são muito custosos para o processador porque necessitam que todas as instruções posteriores ao *misprediction* sejam revertidas e quando o erro de previsão é descoberto, muitas instruções já passaram do *issue stage* e estão no pipeline sendo executadas. Esse problema tende a se tornar cada vez maior, já que a tendência é que os processadores aumentem a sua capacidade de executar instruções em paralelo com o aumento de unidades de execução.

Com o objetivo de promover a melhoria da precisão das previsões de *branches,* foi criada uma competição de *branch prediction,* a *Championship Branch Prediction (CBP)* em parceria com a ISCA (*International Symposium of Computer Architecture*) testando diversos algoritmos para previsão de *branches* propostos pelos participantes.

Na edição de 2014 dessa competição, foi inscrito o algoritmo proposto por Jorge Albericio, Joshua San Miguel, Natalie Enright Jerger, e Andreas Moshovos nomeado de *Wormhole* que consiste de uma melhora em relação ao algoritmo de André Seznec vencedor da edição anterior da competição, ISL-TAGE, para o caso específico de *branches* dentro de loops multidimensionais. Esse tipo de *branches* mostram-se difíceis de prever com algoritmos anteriores, porém com a ideia do *Wormhole,* esse tipo de *branch* possui uma previsibilidade bem alta ao levar em conta correlações com iterações anteriores do loop externo.

2. ISL-TAGE

Antes de falarmos sobre o *Wormhole,* é necessário entender como funciona o ISL-

TAGE proposto por André Seznec. Esse preditor é composto de 3 preditores, o TAGE, SC e Loop predictor.

2.1. TAGE

O *TAgged GEometric history length branch predictor* consiste de um preditor base bimodal e vários preditores que utilizam a história do *branch* para predição. Estes preditores de história, possuem tamanhos diferentes de forma que o tamanho deles formam uma progressão geométrica entre eles. O preditor bimodal trata os casos mais simples onde o *branch* possui uma tendência mais forte para uma direção, enquanto os outros tratam as exceções. O TAGE possui um algoritmo de seleção de qual preditor tomar com base no acerto de cada um.

2.2. Statistical Corrector (SC)

O SC, trata os *branches* que o TAGE não consegue uma boa precisão, mas possuem uma tendência a uma direção ele utiliza *saturating counter tables* e um algoritmo que determina quando ele altera a previsão do TAGE e quando ele a mantém.

2.3. Loop predictor

O *loop predictor* consiste de um preditor simples que determina quando um *branch* representa um loop após ele tomar um padrão semelhante sete vezes. Após isso o *loop predictor* armazena o número de iterações total que o loop executa para que ele sempre acerte nas próximas execuções do mesmo loop.

3. Wormhole

Para descrever como funciona a ideia do algoritmo, podemos analisar o programa da figura 1 abaixo. O programa exemplo representa uma operação em uma matriz feita em dois loops onde apenas há o salto quando o loop interno e externo estão no mesmo número de iteração.

```
// A is the matrix
// B is the right hand side
// X is the current solution estimate
// X0 is the partial solution
for ( i = 0; i < N; i++ ) {   // Loop 3
  X0[i] = B[i];
  for ( j = 0; j < N; j++ )       // Loop 4
    if ( j != i )                  // Branch 2
      X0[i] = X0[i] - A[i + j*n] * X[j];
```

(a)

Figura 2. Iterações do programa exemplo

O algoritmo *Wormhole* proposto consiste de armazenar o histórico do loop externo anterior para estabelecer uma correlação entre o caminho do branch na iteração anterior do loop externo e iteração anterior do loop interno. Esse tipo de abordagem mostra-se eficiente para prever grande parte dos *branches* dentro de loops multidimensionais e estabelece uma visão diferente da correlação utilizada anteriormente que não levava em conta a posição do *branch* dentro de loops.

Podemos ver pela Figura 2, que representa a execução do programa da Figura 1, onde as iterações do loop externo são representados pelas linhas e as iterações do loop interno são representados pelas colunas. A célula cinza mais escuro representa a execução atual do *branch* que deverá ser determinado a direção, enquanto que as células cinza mais claro representam quais execuções do branch serão levadas em contas na correlação para determinar a direção do *branch.* Podemos ver que após a primeira execução representada na Figura 2 (a), o preditor aprende que a correlação "0100" levará a um resultado 1, onde 1 representa *branch taken* e 0 representa *branch not taken.* Assim, na execução da Figura 2 (b), o preditor irá acertar a previsão e nas demais execuções ele irá sempre acertar.

Podemos verificar que para este preditor, é imprescindível que seja conhecido o número de iterações que os loops terão. Isso é resolvido pelo *Loop predictor* que está presente no ISL-TAGE que é utilizado para os demais *branches.*

Para a Figura 2 acima, foi considerado o preditor *Wormhole* de 4kB, que leva em conta apenas os vizinhos mais próximos, porém para o *Wormhole* de 32kB, podemos verificar pela figura 3 abaixo que ele também leva em conta os vizinhos com uma distância de 2 iterações tanto para o loop externo como interno.

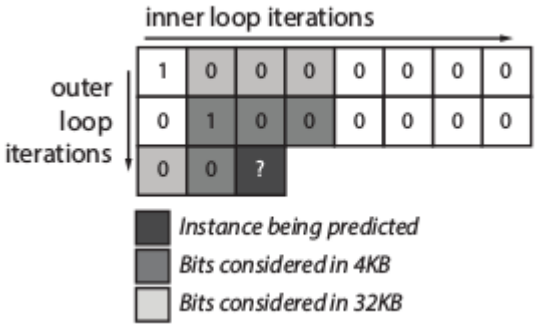

Figura 3. Bits considerados pelos preditores de 4kB e 32kB

4. Materiais e Métodos

Para este trabalho, foi necessário a utilização do simulador da CBP-5 de 2016 além do código *predictor.cc* e *predictor.h* para a categoria de 4kB e 32kB implementados pelos autores do algoritmo *Wormhole* inscrito na CBP-4 com devidas adaptações para o simulador mais recente. Como para a CBP-5 os programas de entrada do simulador não são os mesmos da competição anterior e não foi possível simular o preditor ISL-TAGE por ele ser feito para a CBP-3 que possuía um simulador muito diferente do atual, não foi possível reproduzir exatamente sob os mesmos *traces* do paper *Wormhole: Wisely Predicting Multidimensional Branches".*

Para levantar mais dados, foi simulado então os preditores do *Wormhole* de 4kB e 32kB, foi simulado o ISL-TAGE de 4kB e 32kB baseado no código do Wormhole, retirando o tamanho disponível para o preditor do Wormhole. Foi simulado também um ISL-TAGE de 32kB e 4kB onde não há preditores de histórico no TAGE, apenas o preditor base bimodal. Foi utilizado também um preditor apenas com o elemento bimodal do TAGE de 4kB e 32kB, mas com o Wormhole proposto pelos autores. Vale ressaltar que para o ISL-TAGE que foi representado aqui, não é fiel ao ISL-TAGE implementado por André Seznec, por ter sido implementado pelos autores do paper do Wormhole, e possuir a progressão geométrica que estes escolheram para o preditor deles.

Os *traces* utilizados para todas as simulações foram os LONG-1, LONG-2, LONG-3, LONG-4, LONG-5, LONG-6, LONG-7, LONG-8 e LONG-12 do kit CBP-5. Foram levantados o MPKI (*Misprediction per kilo instruction*), *predictor accuracy,* e a quantidade de *branches,* instruções, *branches* condicionais e incondicionais.

5. Resultados

Os resultados das simulações feitas podem ser vistos abaixo nas tabelas 1 e 2 e na Figura 4.

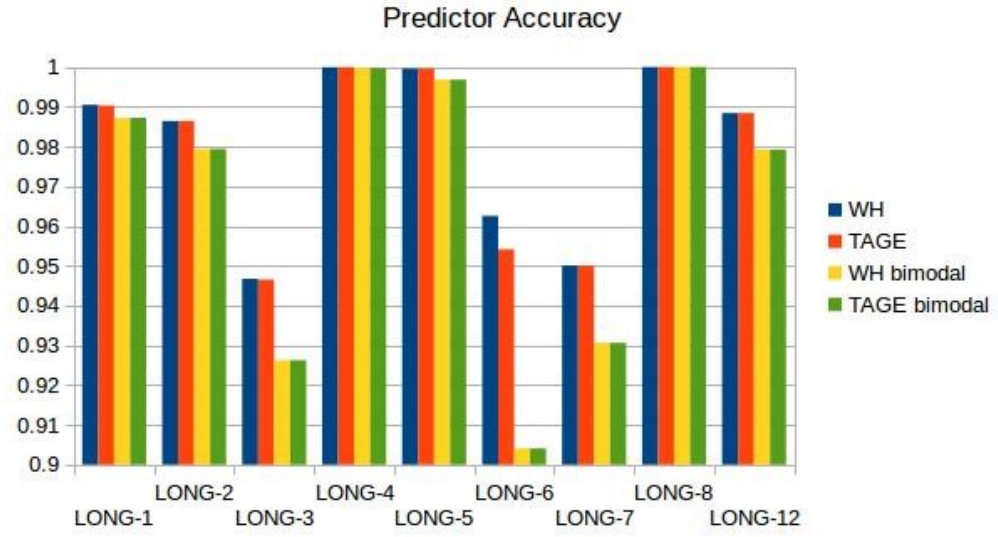

Figura 4. *Predictor Accuracy* dos preditores de 4kB para cada programa.

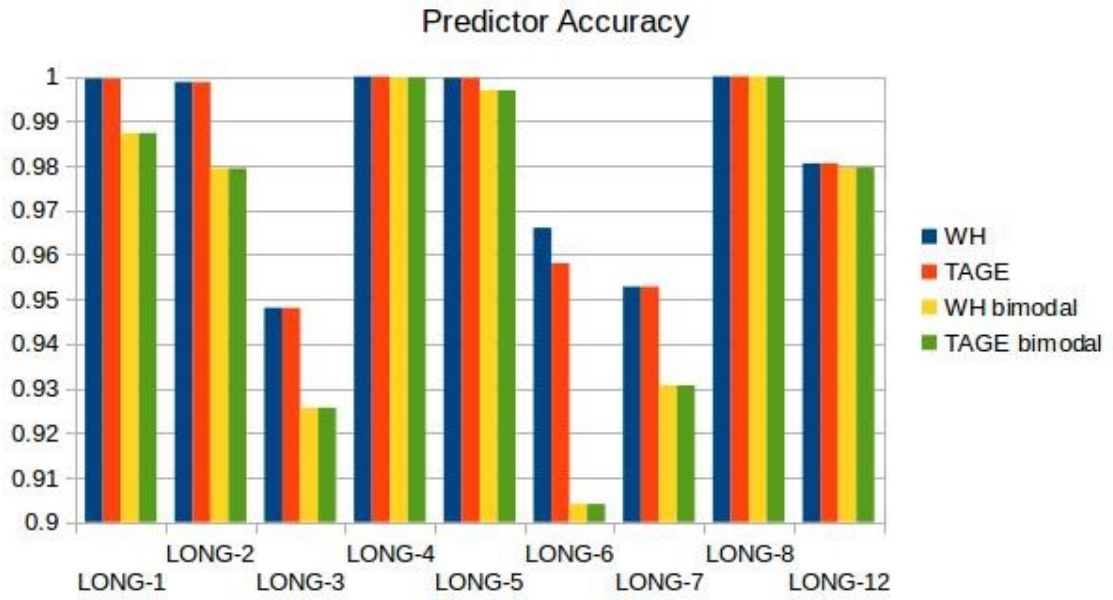

Figura 5. ***Predictor Accuracy*** **dos preditores de 32kB para cada programa.**

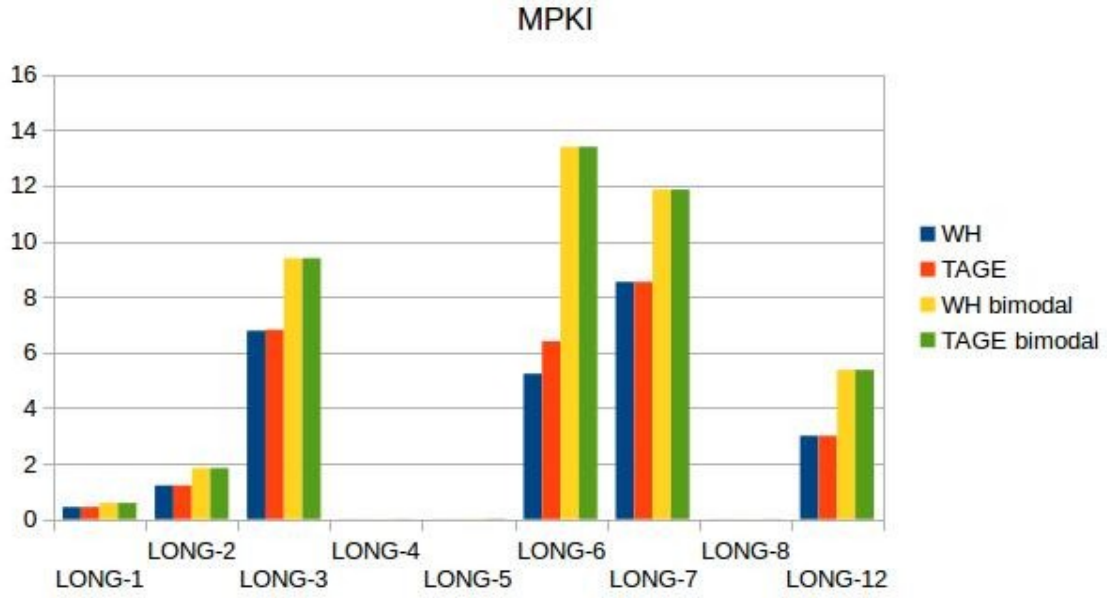

Figura 6. ***MPKI*** **dos preditores de 4kB para cada programa.**

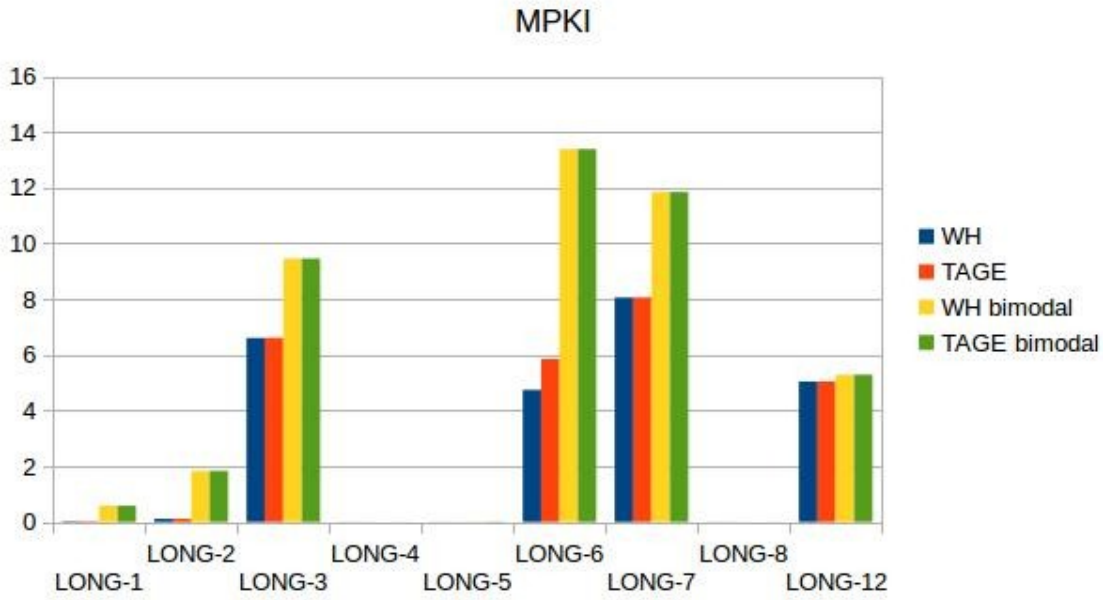

Figura 7. *MPKI* dos preditores de 32kB para cada programa.

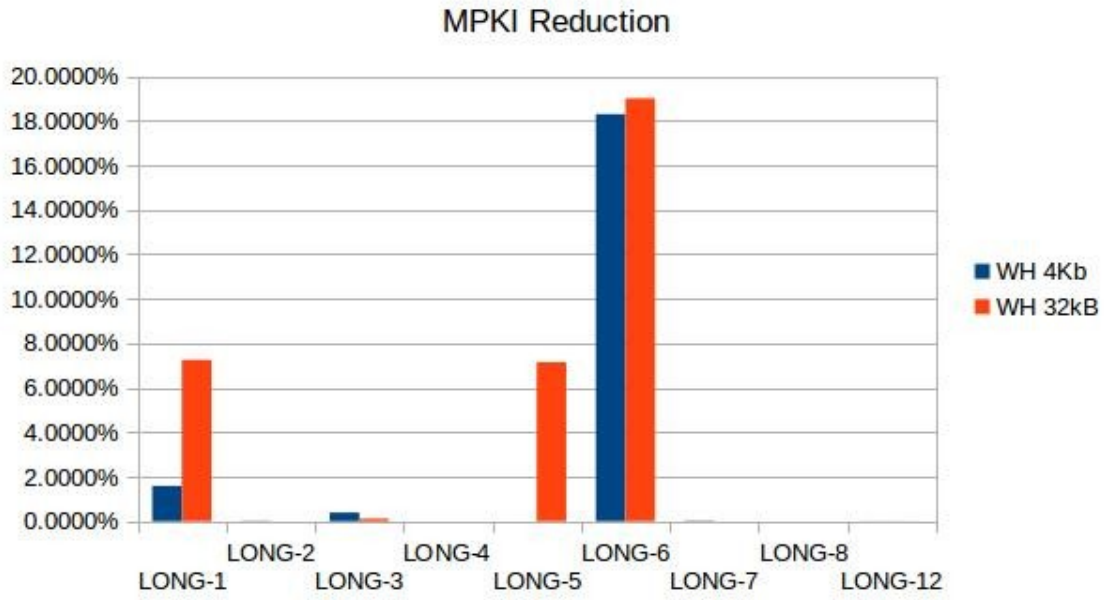

Figura 8. *MPKI reduction* em relação ao ISL-TAGE para cada programa.

References

"Wormhole: Wisely Predicting Multidimensional Branches" Jorge Albericio, Joshua San Miguel, Natalie Enright Jerger, and Andreas Moshovos*Edward S. Rogers Sr., Department of Electrical and Computer Engineering, University of Toronto*

"A 64 Kbytes ISL-TAGE branch predictor" André Seznec, INRIA/IRISA

"A case for (partially) TAgged GEometric history length branch prediction" André Seznec, Pierre Michaud, IRISA/INRIA/HIPEAC