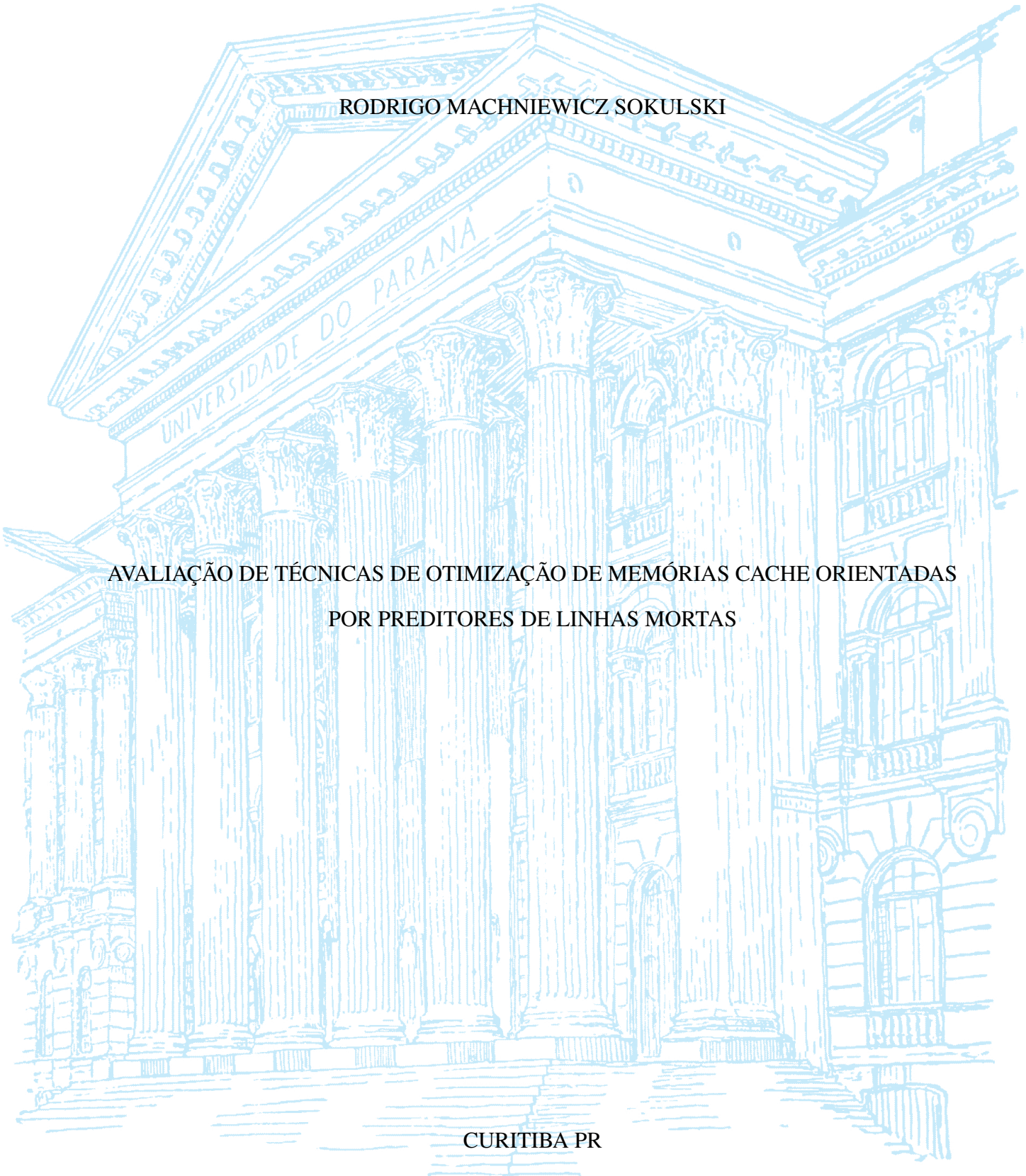
UNIVERSIDADE FEDERAL DO PARANÁ

RODRIGO MACHNIEWICZ SOKULSKI



AVALIAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO DE MEMÓRIAS CACHE ORIENTADAS POR PREDITORES DE LINHAS MORTAS

CURITIBA PR

2019

RODRIGO MACHNIEWICZ SOKULSKI

AVALIAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO DE MEMÓRIAS CACHE ORIENTADAS POR PREDITORES DE LINHAS MORTAS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Marco Antonio Zanata Alves.

CURITIBA PR

2019

A todas as pessoas que, de alguma forma, contribuíram com minha formação.

AGRADECIMENTOS

Aos meus pais e irmão, por todo o suporte e confiança essenciais em tudo o que já realizei.

Ao Prof. Dr. Marco A. Zanata Alves, pela preocupação, apoio, orientação e confiança durante todo o desenvolvimento deste trabalho.

À minha amiga Glenda, pelo seu apoio nos momentos mais difíceis e por suas grandes expectativas que me movem a tentar melhorar sempre.

Aos meus demais amigos e professores por todas as grandes lições durante esta etapa da minha formação.

RESUMO

A constante redução no tamanho dos transistores utilizados na produção de processadores tem levado a um aumento crescente no impacto da energia estática em relação ao consumo energético total destes componentes. Este gasto concentra-se principalmente nas memórias cache de último nível, devido a seu grande tamanho e reduzido número de acessos em relação ao número de linhas desse nível de cache. Diversos mecanismos e técnicas tentam mitigar este problema, sendo que alguns dos mais eficientes, os denominados preditores de linhas mortas, tentam prever os padrões de acesso das caches, para pemitir a aplicação de técnicas de otimização energética e de substituição sobre as linhas de cache. Porém, nem todos os mecanismos utilizam o mesmo conjunto de técnicas. Além disso, algumas técnicas, como o *Drowsy*, não são utilizadas por nenhum preditor. Este trabalho examina o efeito de algumas técnicas de otimização energética e de substituição, avaliando seu desempenho individual, assim como sua influência mútua. Para isso, são avaliadas duas técnicas de otimização energética, o *Gated-Vdd* e o *Drowsy*, demonstrando que, apesar de possuir ganhos em energia estática menores na *Last Level Cache* (LLC), o *Drowsy* consegue ganhos energéticos maiores quando é considerado o impacto de predições errôneas nos demais componentes arquiteturais. Em seguida, são realizados estudos combinando duas técnicas de otimização de substituição, a priorização de linhas mortas e o *Bypass*, com a técnica de otimização energética *Gated-Vdd*, de forma a avaliar a existência de uma melhor combinação destas técnicas para cada mecanismo preditor testado. Os resultados obtidos sugerem que as técnicas de otimização não são universais, ou seja, diante de um novo mecanismo de predição, devem existir estudos sobre quais as melhores técnicas a se aplicar de forma a obter bons ganhos, tanto energéticos quanto em relação ao desempenho.

Palavras-chave: Memória cache. Preditores de linhas mortas. Otimização de cache.

ABSTRACT

The constant reduction in the size of transistors used in the production of processors has led to a growing increase in the importance of the static energy in relation to the total energy consumption expended by these components. This expense is mainly concentrated in the last level cache memories, due to their large size and the small number of accesses relative to the number of rows of this cache level. Several mechanisms and techniques try to mitigate this problem, and some of the most effective, the so-called dead line predictors, try to predict the cache access patterns, to allow the application of energetic and replacement optimization techniques over the cache lines. However, not all mechanisms use the same set of techniques. Furthermore, some techniques, such as Drowsy, are not used by any predictor. This work examines the effect of some energetic and replacement optimization techniques, evaluating their individual performance as well as their mutual influence. For this, two energetic optimization techniques are evaluated, Gated-Vdd and Drowsy, demonstrating that, despite having lower static energy gains in the Last Level Cache (LLC), Drowsy achieves higher energy gains when considering the impact of erroneous predictions on other architectural components. After that, studies are conducted combining two substitution optimization techniques, dead line prioritization and Bypass with the energy optimization technique Gated-Vdd, to evaluate the best combination of these techniques for each predictor mechanism tested. The obtained results suggest that the optimization techniques are not universal, that is, in face of a new prediction mechanism, there must be studies on which the best techniques to apply in order to obtain good gains, both in energy and performance.

Keywords: Cache memory. Dead line predictors. Cache optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AHT *Access History Table*.

AIP *Access Interval Predictor*.

DEWP *Dead Line and Early Write-Back Predictor*.

DRAM *Dynamic Random Access Memory*.

DSBP *Dead Sub-Block Predictor*.

LLC *Last Level Cache*.

LvP *Live-time Predictor*.

SDP *Skewed Dead Block Predictor*.

SUMÁRIO

1 INTRODUÇÃO

Ao pensar em desempenho, de sistemas computacionais, o primeiro componente que vem à mente é o processador, responsável por quase todas as operações dos sistemas atuais. Porém, para sua execução, operações consomem dados de entrada e geram dados de saída, que são armazenados nas memórias da máquina. Com a velocidade dos processadores aumentando, em média, 21% ao ano (Jeff Preshing, 2012), o volume de dados gerados e consumidos tende a seguir a mesma tendência. Apesar disso, de acordo com Chang (2017), o aumento anual médio no desempenho das DRAMs tem sido de apenas 1.5%. Essa diferença de rendimento cria um gargalo nos sistemas, gastando grande parte do tempo disponível para processamento com a transmissão de dados entre memória principal e processadores.

A maior parte dos acessos à memória segue os princípios de localidade espacial e temporal. O princípio da localidade espacial define que, após um acesso à memória, o próximo acesso tem grandes chances de ocorrer na mesma região de memória do anterior. Exemplos desse princípio são acessos sequenciais a vetores e o uso de variáveis internas em funções. Já o princípio da localidade temporal define que regiões de memória acessadas a menos tempo têm uma chance maior de receberem os próximos acessos. As chamadas de funções são exemplos deste princípio. Após uma chamada de função, suas variáveis internas receberão diversos acessos antes da função retornar e, somente após esse retorno, as variáveis no escopo de seu chamador poderão ser acessadas novamente.

Baseadas nesses dois princípios, memórias de acesso mais rápido e menor capacidade de armazenamento (caches) são adicionadas entre os processadores e a memória principal como forma de mitigar o gargalo entre eles. Esses novos componentes seguem os princípios de localidade espacial e temporal para armazenar os dados com maior probabilidade de reutilização, assim reduzindo os acessos à memória principal do sistema e acelerando-o.

Tabela 1.1: Configurações de hierarquia de memória.

Nome	Tamanho	Associatividade	Tempo de acesso (ciclos)
L1	32~KB	8 Ways	4
L2	256~KB	8 Ways	8
L3	4~MB	16 Ways	26
RAM	1~GB	–	200

A Figura 1.1 mostra a percentagem de tempo adicional necessário para a execução de diversas aplicações em um sistema onde as caches da hierarquia de memória da Tabela 1.1 foram retiradas. A utilização das caches reduziu o tempo médio de execução em 94%, sendo que o desempenho de todas as aplicações aumentou em mais de 75%. Esses ganhos demonstram a importância das caches para os sistemas atuais e são decorrentes da alta latência de acesso aos dados da memória principal, que é em torno de 10 vezes mais lenta que o último nível de cache.

Devido às suas vantagens, com o tempo as caches cresceram. Atualmente ocupam aproximadamente 50% do chip dos processadores e detêm um consumo energético considerável, algumas vezes superando 40% do consumo energético total desse chip, como mostrado em Alves (2014). Este gasto das caches tem duas fontes: a energia utilizada para fazer *gating* dos circuitos durante leituras ou escritas, denominada energia dinâmica, e a energia utilizada para manutenção dos conteúdos em memória, denominada energia estática.

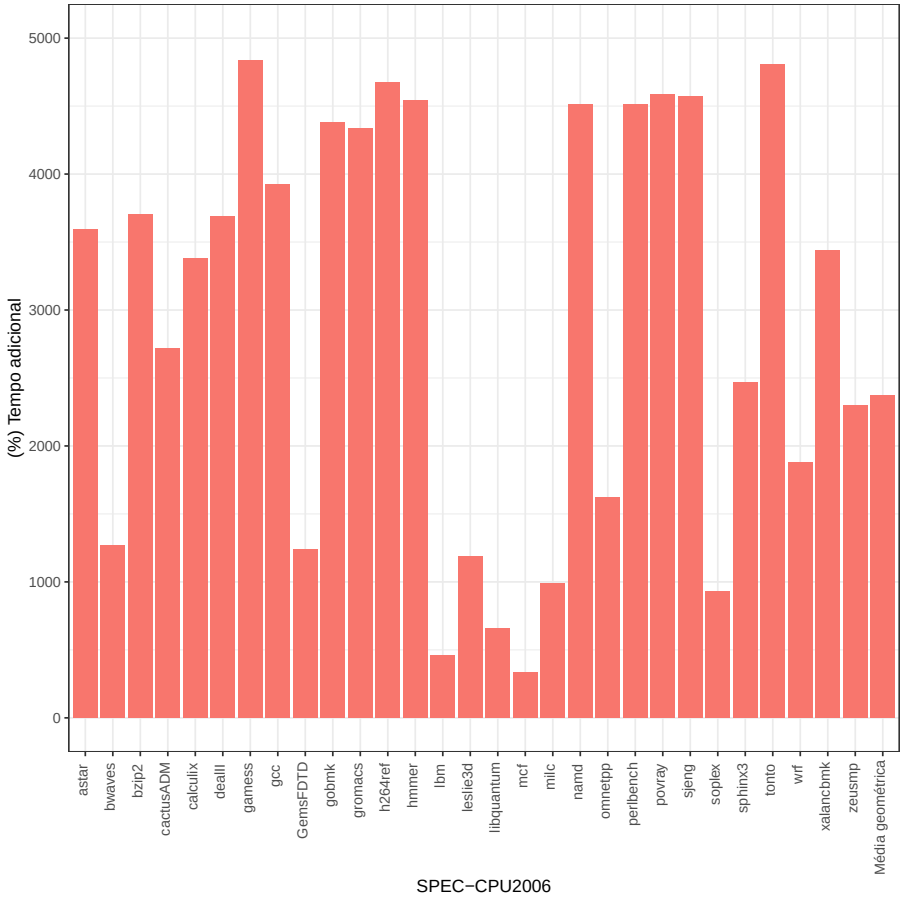


Figura 1.1: Percentagem de tempo de execução adicional sem o uso de memórias cache, em relação ao tempo de execução com a hierarquia de caches da Tabela 1.1. A execução com a hierarquia de caches é representada pelo 0% de cada coluna. Quanto maior uma coluna, mais o tempo de execução aumentou com a retirada das caches. Assim, se uma coluna chegou a 1000% o tempo de execução da aplicação representada por ela aumentou 10 vezes ao serem retiradas as caches.

A evolução dos componentes eletrônicos, a redução no tamanho dos transistores e a consequente diminuição de sua tensão permitiram uma contenção no consumo de energia dinâmica dos processadores, visto que esse tipo de energia é proporcional ao quadrado da tensão utilizada no sistema (Kim et al., 2003). Porém, o crescimento das memórias e a redução do tamanho e tensão de seus transistores têm potencial para aumentar de forma crítica o consumo de energia estática (Zhou et al., 2003), que pode crescer em mais de 5 vezes a cada nova geração de processadores, como previsto por Borkar (1999). Visto que esse tipo de energia, somente das caches, já representa quase 7% de todo o consumo energético de alguns processadores (Alves, 2014), o controle desse gasto é essencial.

Grande parte do consumo de energia estática dos processadores é usado na manutenção dos conteúdos em suas caches. Mas nem sempre esse gasto é justificável. A Figura 1.2 mostra a percentagem média do tempo de vida de uma linha de cache da LLC após ter recebido seu último acesso. Durante esse período, denominado morte da linha, que chega a 50% em grande parte das aplicações, toda a energia estática gasta em sua manutenção é desperdiçada, visto que, como a linha não será mais acessada, seu conteúdo poderia ser mantido na memória principal do sistema. Apesar desse grande desperdício energético com linhas mortas, o último nível de cache desempenha um papel importante para a execução de muitas aplicações, visto que sua

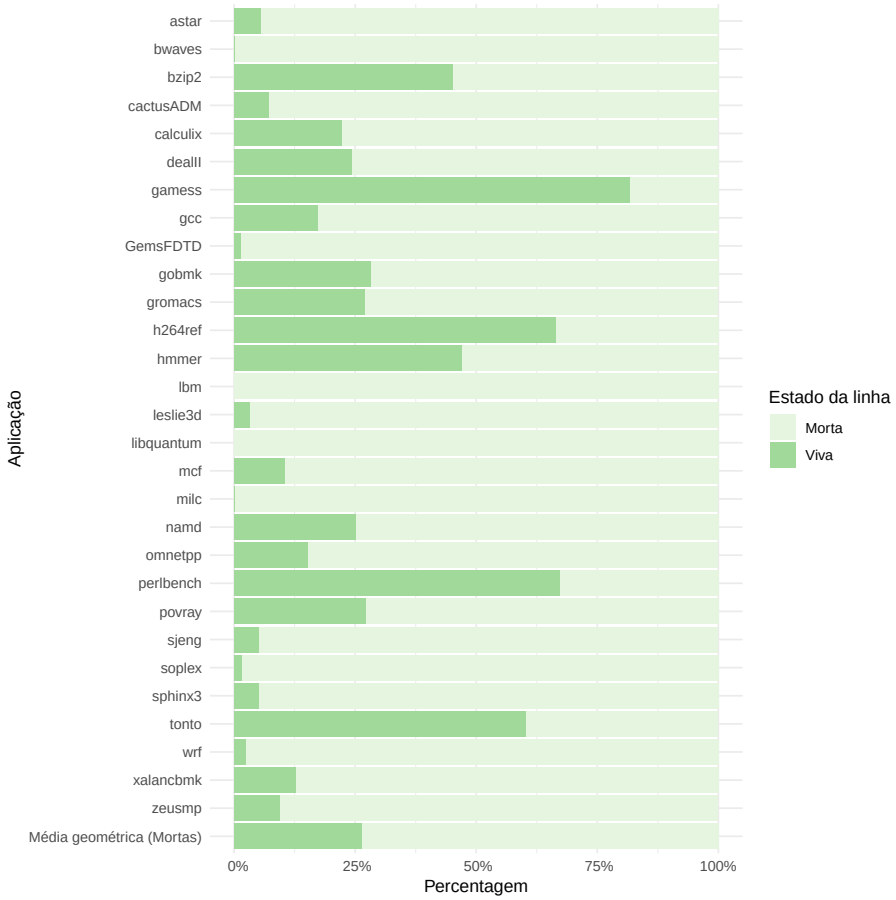


Figura 1.2: Percentagem média do tempo durante o qual as linhas de cache no último nível da hierarquia da Tabela 1.1 estão mortas.

remoção pode levar a degradações de mais de 30% no tempo de execução desses programas, como é mostrado em Santos et al. (2016).

Diversos mecanismos tentam minimizar o desperdício de energia estática das caches. Alguns tentam prever quando linhas recebem seu último acesso para então aplicar políticas de otimização sobre elas. Esses mecanismos são denominados preditores de reúso (*cache reuse predictors*) ou preditores de linhas mortas (*cache dead line predictors*).

Cada preditor de reúso pode adotar um conjunto de políticas de otimização diferente. Dentre as diversas políticas que podem ser adotadas, algumas tentam aumentar a eficiência das caches, como a priorização de linhas mortas para substituição e o *bypass* de linhas *dead-on-arrival*, que evita que dados que receberão apenas um acesso sejam instalados. Além dessas, podem ser aplicadas políticas de redução direta de energia em linhas mortas, como as técnicas *Gated-Vdd* e *Drowsy*, propostas por Powell et al. (2000) e Flautner et al. (2002), a fim de economizar a energia estática utilizada na manutenção de linhas mortas.

Este trabalho visa investigar os possíveis benefícios da combinação de diferentes políticas de otimização da substituição com diferentes técnicas de economia energética direta, aplicadas por preditores de linhas mortas.

1.1 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho está organizado da seguinte maneira: o capítulo 2 aborda mecanismos preditores de linhas mortas de cache, apresentando quais propriedades da cache utilizam para

suas previsões e quais das políticas de otimização avaliadas neste trabalho são empregadas por cada um. Em seguida, o capítulo 3 apresenta a metodologia aplicada para a realização deste trabalho. No capítulo 4 são apresentados os mecanismos e técnicas utilizados nas avaliações dos capítulos 5 e 6. O capítulo 5 compara o desempenho de duas técnicas de otimização energética, o *Gated-Vdd* e o *Drowsy*, sobre os preditores *Dead Line and Early Write-Back Predictor* (DEWP) e *Skewed Dead Block Predictor* (SDP). Já o capítulo 6 faz uma avaliação das interferências entre a técnica de redução energética *Gated-Vdd* e duas políticas de otimização de desempenho, a priorização de linhas mortas e o *Bypass*. A conclusão deste trabalho é apresentada no capítulo 7.

2 MECANISMOS RELACIONADOS

A busca por linhas mortas de cache não é um problema novo. Nas últimas décadas, diversos mecanismos foram criados, cada um utilizando uma ou mais propriedades estáticas da cache.

Este capítulo apresenta as propriedades utilizadas por alguns destes mecanismos e mostra, na Tabela 2.1, quais das otimizações de cache estudadas neste trabalho foram utilizadas por cada um.

2.1 MECANISMOS E PROPRIEDADES

***Skewed Dead Block Predictor* (SDP)**, proposto por Khan et al. (2010), utiliza o endereço das instruções que acessam cada bloco da cache para prever sua morte. Muitas vezes, uma mesma sequência de instruções realiza todos os acessos recebidos por diversos blocos, como em laços usados para processar elementos de listas. Nesses laços, cada elemento da lista só é acessado durante uma de suas iterações. Assim, após a execução da última instrução dessa iteração, o bloco está morto. Identificando a última instrução dessa sequência contida no laço, o mecanismo pode estimar a morte de um bloco cada vez que essa instrução é executada.

Hawkeye, proposto por Jain e Lin (2016), tenta reconstruir a política de substituição ótima da cache, denominada algoritmo de Belady. O algoritmo de Belady baseia-se na observação de eventos futuros da cache para determinar quais as melhores políticas de gerenciamento de suas linhas. Para mimetizar este comportamento, *Hawkeye* aplica o mesmo algoritmo a eventos passados e, assumindo que seu comportamento se repetirá, obtém resultados bem próximos aos de uma aplicação teórica do verdadeiro.

***Access Interval Predictor* (AIP)**, proposto por Kharbutli e Solihin (2008), contabiliza o número de acessos recebido por cada conjunto da cache. Com essa informação, o mecanismo descobre o número médio de acessos que o conjunto de cada bloco receberá entre dois acessos a esse bloco. Caso o número de acessos recebido pelo conjunto de um bloco ultrapasse esse número médio, o bloco pode ser previsto como morto.

***Live-time Predictor* (LvP)**, proposto por Kharbutli e Solihin (2008), contabiliza o número médio de acessos que cada bloco da cache recebe durante sua vida. Sabendo quantos acessos uma linha espera receber, esse mecanismo pode predizer a linha como morta após esse número de acessos ser atingido.

Leeway, proposto por Faldu e Grot (2017), contabiliza quantos blocos são acessados em cada conjunto da cache, ignorando múltiplos acessos ao mesmo bloco. Assim, mesmo que um bloco seja acessado diversas vezes, esse múltiplos acessos serão contabilizados como um único acesso a bloco. Com isso, para cada linha de cache, o mecanismo tenta descobrir quantos blocos de seu conjunto são acessados entre dois de seus acessos. Caso o número de blocos acessados no conjunto de uma linha ultrapasse esse número, a linha pode ser prevista como morta.

Perceptron, proposto por Teran et al. (2016), baseia-se na aprendizagem dos perceptrons (Block, 1988) para utilizar endereços de instruções e regiões de memória em suas predições. Esse mecanismo combina as últimas instruções que acessaram cada bloco e sua região de memória para atribuir uma probabilidade de morte a esse bloco. Assim, quando um bloco atinge uma probabilidade superior a um limiar predefinido é considerado morto.

***Dead Sub-Block Predictor* (DSBP)**, proposto por Alves et al. (2012), contabiliza o número de acessos recebidos por cada sub-bloco da cache durante sua vida. Assim, caso um sub-bloco atinja seu número previsto de acessos, pode ser declarado morto.

***Dead Line and Early Write-Back Predictor* (DEWP)**, proposto por Alves et al. (2013), descobre quantos acessos um bloco espera receber, de acordo com a instrução que realizou seu primeiro acesso e com o *byte* acessado por essa instrução. A partir desse número esperado de acessos, o mecanismo consegue identificar a morte de cada bloco logo após seu último acesso.

Tabela 2.1: Políticas de otimização de cache aplicadas pelos preditores de linhas mortas apresentados na seção 2.1.

Mecanismo	Priorização de linhas mortas	Bypass	Método de economia energética
SDP	✓	✗	✗
Hawkeye	✓	✗	✗
AIP	✓	✓	✗
LvP	✓	✓	✗
Leeway	✓	✓	✗
Perceptron	✓	✓	✗
DSBP	✓	✗	Gated-Vdd
DEWP	✓	✗	Gated-Vdd

2.2 CONCLUSÃO

Este capítulo apresentou as propriedades da cache utilizadas por alguns dos mecanismos preditores de reúso presentes na literatura. Grande parte deles utiliza apenas métodos de otimização de desempenho aliados a suas previsões, sem considerar técnicas aplicadas para redução de energia estática. Além disso, o conjunto de técnicas utilizadas varia entre os preditores, sugerindo existir possibilidade de melhorias com sua variação. Os preditores utilizados nas avaliações deste trabalho, *Skewed Dead Block Predictor* (SDP) e *Dead Line and Early Write-Back Predictor* (DEWP), foram escolhidos por serem os dois preditores melhor avaliados em Alves (2014), para uso na *Last Level Cache* (LLC), e serão detalhados no capítulo 4.

3 METODOLOGIA

O presente trabalho consiste de uma pesquisa descritiva (Gil, 2002), buscando estabelecer correlações entre diversas técnicas aplicadas a preditores de reúso e melhorias na eficiência energética da *Last Level Cache* (LLC). Por meio de análises experimentais, essas diversas técnicas são avaliadas em experimentos controlados. Os resultados dos experimentos são utilizados para verificar possíveis relações entre mecanismos preditores de reúso, suas políticas de otimização e a performance da LLC.

Como mostrado no capítulo 2, diversos mecanismos preditores de linhas mortas de cache existem na literatura. Esses preditores estimam quando cada conteúdo em cache não será mais acessado para aplicar alguma política de otimização sobre ele. Das diversas políticas de otimização, apenas um conjunto pequeno é utilizado por cada preditor, sendo que, para todos os mecanismos encontrados durante a pesquisa bibliográfica, o conjunto utilizado é similar.

Quando se trata de economia energética, a técnica *Gated-Vdd* é a mais utilizada. Essa técnica desliga linhas de cache mortas para economizar sua energia, perdendo seu conteúdo. Uma alternativa para ela é a *Drowsy*, que apenas reduz a voltagem das linhas, mantendo seu conteúdo. Porém, essa variação não é implementada em nenhum preditor encontrado durante a pesquisa bibliográfica. Outra consideração é que cada preditor de reúso é baseado em atributos diferentes de cada acesso à memória. Esses diferentes atributos levam a diferentes predições, que de acordo com as políticas de otimização adotadas, alteram o comportamento da cache. Essas alterações podem mudar o estado futuro do sistema, gerando novos misses e alterando as previsões futuras.

Visto que apenas um conjunto pequeno de políticas é adotado por diversos preditores, este trabalho busca avaliar se as políticas mais frequentemente adotadas pelos preditores de reúso são universais, ou seja, se qualquer bom preditor que as adote obterá altos ganhos para seu sistema. Como as linhas previstas como mortas variam entre os preditores, existem chances de que uma política muito utilizada não seja a mais adequada em determinados casos. Para avaliar isso, os experimentos deste trabalho se dividem em duas etapas. Primeiro a técnica de redução energética mais utilizada nos preditores de reúso (*Gated-Vdd*) e a técnica *Drowsy* são comparadas. Esse paralelo busca encontrar situações onde a *Drowsy* apresenta maiores ganhos que a *Gated-Vdd*, justificando seu uso. Em seguida, são avaliadas duas técnicas de otimização adotadas com frequência nos preditores de linhas mortas, a Prioridade e o *Bypass*. Essa análise busca encontrar quais os ganhos fornecidos por cada técnica e o quanto ela influencia no resultado final do mecanismo.

Todos os testes realizados para este estudo foram implementados sobre um simulador próprio escrito em C++. O simulador é capaz de operar em dois modos. O primeiro utiliza traços obtidos pelo PinPoints (Patil et al., 2004) que representam as parcelas mais significativas das intruções de cada aplicação. O segundo utiliza o instrumentador binário dinâmico Pin (Intel, 2018) para simular todos os acessos à memória de cada aplicação.

Como o capítulo 5 avalia o desempenho de técnicas de redução energética, as transições entre regiões com muitos e poucos acessos à memória são essenciais em seus resultados. Isso ocorre porque, nas regiões de pouco acesso, substituições demoram a ocorrer, assim o desligamento ou não de uma linha de cache pode afetar significativamente a capacidade de economia de uma combinação de mecanismo e técnica. Portanto, os experimentos desse capítulo utilizaram a instrumentação binária dinâmica para obter uma cobertura completa das instruções acessadas por cada aplicação.

Os experimentos do capítulo 6 têm como foco a otimização dos conteúdos presentes em cache. Nesse caso, os benefícios das técnicas avaliadas não são aproveitados por códigos com pouca interação com a memória. Assim, a utilização dos traços contendo apenas as instruções mais significativas de cada aplicação torna-se mais atrativa, visto que avalia diretamente as regiões críticas para as otimizações, além de possuir um menor tempo de simulação.

Todos os experimentos deste trabalho foram realizados sobre as 29 aplicações do SPEC-CPU 2006, que é um conjunto de benchmarks representando cargas de trabalho de aplicações reais. Além disso, os estudos foram feitos sobre a cache de último nível, devido ao seu maior consumo energético e à sua grande latência no caso de misses. Esses maiores custos evidenciam os efeitos causados pela redução energética e por predições incorretas dos mecanismos analisados. Em cada etapa das avaliações, uma configuração diferente da hierarquia de memória foi utilizada, sendo especificada na seção de sua utilização. Essa variação de configurações foi causada pelas validações realizadas entre os mecanismos implementados neste trabalho e suas descrições presentes em seus trabalhos originais. As validações do simulador foram realizadas em duas etapas: primeiramente foram efetuadas validações manuais para verificar o funcionamento da simulação das caches e da *Dynamic Random Access Memory* (DRAM) e, já com os preditores implementados, foram realizadas validações com base nos resultados obtidos pelo simulador e nos dados fornecidos pelos trabalhos originais de cada preditor. O consumo de energia dos componentes destas configurações foi obtido com o programa CACTI 6.5 (Muralimanohar et al., 2008). Os experimentos realizados neste trabalho não possuem indeterminismos, visto que o sistema operacional não foi simulado e as aplicações simuladas foram produzidas por compilações estáticas. As limitações presentes na metodologia utilizada neste trabalho são descritas a seguir.

3.1 LIMITAÇÕES

Os experimentos realizados neste trabalho são baseados em um simulador desenvolvido durante essa pesquisa, modelando apenas a hierarquia de memória. Durante sua execução, esse simulador ignora os efeitos da execução fora de ordem, a qual poderia mascarar a latência dos acessos à memória principal. A utilização de um *pipeline* também não foi modelada, assim, considerando que as instruções não podem ser executadas com IPC maior que 1. Além disso, como a DRAM não foi implementada detalhadamente, todos os acessos a ela levam tempo constante. Foi considerado que instruções que não realizam acessos à memória são executadas em um único ciclo, ignorando sua real latência e reduzindo seu impacto nos cálculos de energia e tempo. Por fim, como nos experimentos realizados, os barramentos não foram considerados nos cálculos envolvendo energia, o custo energético dos acessos à DRAM foi reduzido.

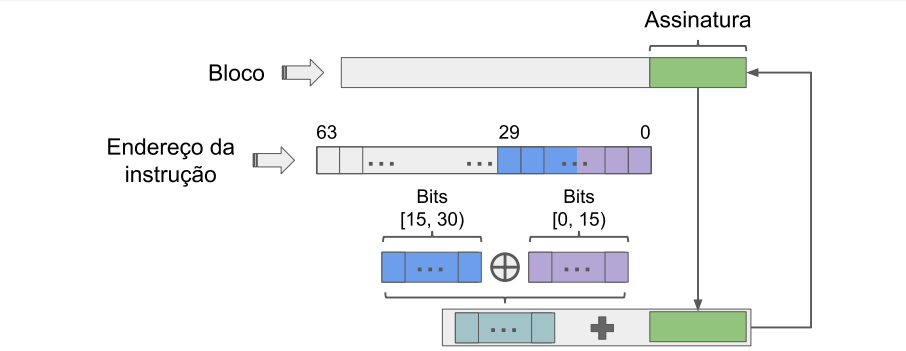


Figura 4.1: Atualização da assinatura do SDP presente em um bloco de cache, durante um acesso a esse bloco.

4 IMPLEMENTAÇÕES

Para a realizar os experimentos contidos neste trabalho, alguns mecanismos preditores de reúso e políticas de otimização foram implementados, sendo descritos neste capítulo.

4.1 PREDITORES IMPLEMENTADOS

Os experimentos deste trabalho foram realizados sobre os mecanismo preditores de linhas mortas de cache descritos a seguir.

4.1.1 SDP

O mecanismo *Skewed Dead Block Predictor* (SDP), proposto por Khan et al. (2010), é baseado na seguinte ideia: se uma sequência de instruções leva ao último acesso de um bloco de cache, essa mesma sequência faz o mesmo a outros dados. Assim, quando uma dessas sequências "mortais" acessa uma linha de cache, o conteúdo dessa linha pode ser considerado morto. Para descobrir as sequências "mortais" de um programa, o SDP rastreia todas as instruções que acessam cada bloco da cache. Porém, armazenar diretamente um número grande de instruções é inviável. Assim, cada sequência é convertida em uma assinatura de 15 bits que é mantida junto com a linha de cache que a gerou. Quando um novo conteúdo é levado para a cache, a assinatura presente em sua linha é limpa, sendo atualizada a cada acesso recebido pelo bloco. Essa atualização é ilustrada pelo Figura 4.1 e ocorre da seguinte forma:

1. Os 30 bits menos significativos do endereço da instrução acessando o bloco são divididos em duas partes de 15 bits.

2. Uma disjunção exclusiva (XOR) é feita entre as duas partes.

3. O resultado da disjunção é adicionado ao valor presente na assinatura do bloco.

4. Os 15 bits menos significativos dessa soma tornam-se a nova assinatura.

Quando o conteúdo de um bloco é substituído, sua assinatura contém uma sequência "mortal" que pode ser utilizada para as previsões do mecanismo. Porém, durante a substituição, essa assinatura precisa ser eliminada para dar lugar à próxima sequência. Para salvar o conhecimento obtido a partir de cada assinatura, o SDP conta com duas tabelas de predição. Cada

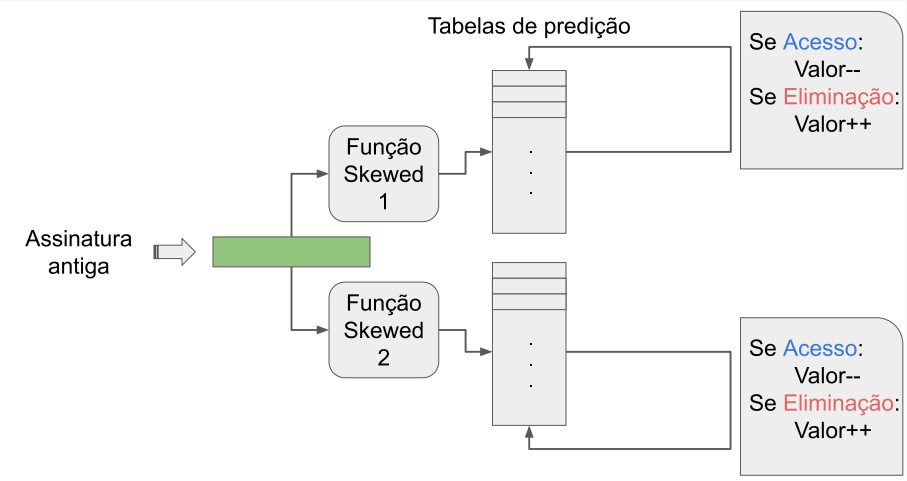


Figura 4.2: Treinamento de tabelas de predição do SDP, durante acessos e eliminações de linhas da cache.

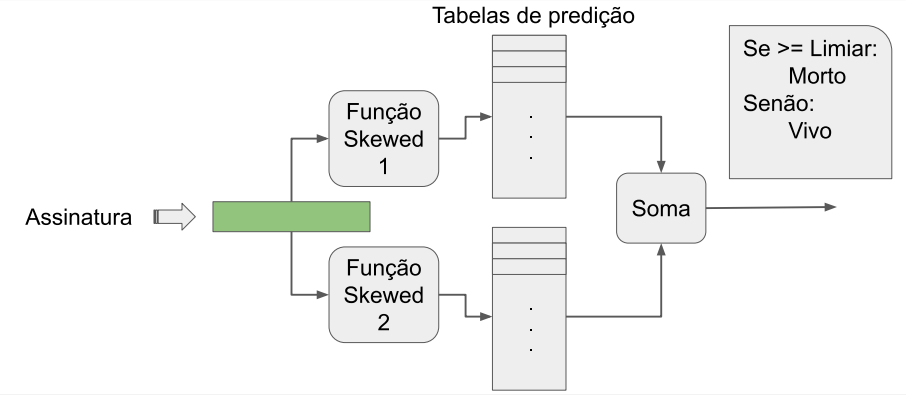


Figura 4.3: Predição do SDP sobre estado de um bloco, baseada na assinatura dos acessos recebidos por esse bloco e realizada após cada acesso às linhas de cache.

tabela é composta por diversos contadores saturados de 2 bits e cada assinatura está vinculada a um destes contadores. Essa ligação é realizada a partir das funções *skewed* propostas por Seznec (1993). Essas funções convertem cada assinatura em dois índices, um para cada tabela de predição. Assim, durante a execução de um programa, as tabelas de predição são treinadas pelas assinaturas de cada bloco da cache, como ilustrado pela Figura 4.2, de acordo com as seguintes regras:

- Durante o acesso a uma linha da cache, sua assinatura é usada para indexar as tabelas de predição e decrementar seus contadores, antes de ser atualizada.
- Durante a substituição do conteúdo de um bloco, sua assinatura é usada para indexar as tabelas de predição e incrementar seus contadores, antes de ser eliminada.

Com isso, contadores relacionados a assinaturas de blocos mortos acumularão altos valores. Já aqueles vinculados a assinaturas de sequências que não receberam seu último acesso tenderão a 0.

Detendo informações sobre as assinaturas que levam blocos à morte nas tabelas de predição, o mecanismo pode definir quando o conteúdo de cada linha de cache recebe seu último acesso para aplicar políticas de otimização. Para isso, a cada acesso recebido por um bloco, o SDP consulta suas tabelas de predição tentando determinar se seu conteúdo ainda está vivo. Essa verificação é ilustrada pela Figura 4.3 e segue os seguintes passos:

1. Com a assinatura atualizada pelo acesso atual, as tabelas de predição são consultadas. O valor de cada tabela é somado e o resultado da soma é comparado a um limiar.

2. Caso essa soma seja maior ou igual ao limiar, o conteúdo é considerado morto. Caso contrário, é dado como vivo.

O valor 2 foi utilizado como limiar por prover uma rápida transição entre as predições de vida e morte das linhas de cache. Para armazenar as previsões do mecanismo, cada bloco contém uma entrada adicional de um bit, que recebe o valor 1 caso seu conteúdo seja previsto como morto, assim identificando as mortes para as políticas de otimização. A partir dessa implementação, esse mecanismo obtém uma média geométrica de erros em suas previsões de 4% para as aplicações do SPEC-CPU 2006.

4.1.2 DEWP

O mecanismo *Dead Line and Early Write-Back Predictor* (DEWP), proposto por Alves et al. (2013), baseia-se na ideia de que diversos blocos, que possuem o mesmo número de acessos, foram trazidos para dentro da cache por uma mesma instrução, que acessou um mesmo *byte* dentro da linha de cache. Um exemplo simples dessa ideia é um loop que itera sobre uma lista executando um pequeno conjunto de operações sobre cada um de seus elementos, como mostrado no algoritmo 4.1.

Algoritmo 4.1: Pseudocódigo acessando um dos elementos de uma lista a cada iteração de um laço.

```
1 for elemento in lista:
2   elemento <- elemento + 1
3   aux <- global_C / elemento
4   if(aux > 15):
5     elemento <- 1
6   else:
7     elemento <- 0
```

Nesse algoritmo existem duas instruções que podem fazer o último acesso a cada elemento da lista. Porém, o número de acessos que cada elemento recebe é sempre o mesmo. Esse número constante de acessos possui uma forte relação com a primeira instrução a acessar cada endereço, já que loops distintos podem possuir padrões diferentes de acesso. Porém, dentro de cada laço esse valor é geralmente constante. Assim, o *Dead Line and Early Write-Back Predictor* (DEWP) combina o endereço dessa instrução e o endereço do bloco acessado para realizar suas previsões.

Quando novos dados são instalados em um bloco, este mecanismo prevê seu futuro número de acessos por meio de uma estrutura denominada *Access History Table* (AHT) e armazena essa informação junto ao bloco em um contador de acessos esperados. A cada novo acesso a esse bloco, seu contador de acessos esperados é decrementado, chegando a 0 no momento da morte de seu conteúdo. Além do contador de acessos esperados, diversos outros metadados são adicionados a cada linha de cache para controlar as previsões do DEWP. Todos os metadados adicionados são descritos a seguir:

- Treino: bit indicando se os acessos recebidos pela linha estão sendo utilizados para treinar a AHT.

- Acessos: contador saturado de 2 bits indicando o número esperado de acessos restantes até a morte do bloco.

- *Overflow*: caso o número esperado de acessos seja maior que o máximo valor armazenado pela entrada “Acessos”, este bit recebe o valor 1. Desta forma, linhas com um número de acessos que não pode ser contabilizado nunca são consideradas mortas.

- Ponteiro: aponta para uma entrada da AHT sendo treinada ou ajustada pelo bloco.

A *Access History Table* (AHT) armazena o número de acessos esperado para cada combinação [instrução, bloco] conhecida. Ela é composta por 512 entradas organizadas de acordo com um mapeamento associativo por conjunto, em 64 conjuntos de associatividade 8. Cada uma de suas entradas possui os seguintes campos:

- PC: entrada de 16 bits contendo a parte menos significativa do endereço da instrução que causou o miss durante o qual a linha da AHT foi alocada.

- Deslocamento: entrada de 3 bits contendo a parte mais significativa do deslocamento do bloco acessado durante a alocação desta linha.

- Vínculo: apenas uma linha de cache pode estar vinculada a cada linha da AHT para treinamento ou ajuste ao mesmo tempo. Este bit indica se já existe esse vínculo.

- Acessos: contador saturado de 2 bits indicando o número de acessos esperados para esta combinação de PC e deslocamento.

- *Overflow*: Caso o número de acessos seja maior que 4, este bit assume valor 1.

Para obter uma previsão da AHT sobre determinado bloco, a instrução que o levou à cache e o offset de seu endereço (*offset*) indexam a AHT da seguinte forma: sendo 0 o bit menos significativo do endereço da instrução, seus bits de 4 a 6 são selecionados. Esses bits são concatenados com os 3 bits mais significativos de *offset* e o resultado da concatenação é usado como índice para os conjuntos da AHT. A Figura 4.4 ilustra a criação desse índice. Os componentes do índice foram escolhidos por proporcionarem uma grande cobertura dos padrões apresentados nos acessos às caches, mantendo uma boa precisão mesmo em pequenas AHTs. Os 3 primeiros bits do *offset* são ignorados porque a maior parte dos acessos à cache ocorre alinhada a blocos de 8 *bytes*. Já os bits de 4 a 6 do endereço da instrução fornecem uma boa granularidade para as instruções, visto que combinam informações mais abrangentes de sua região de memória (índice) com uma diferenciação mais específica (deslocamento).

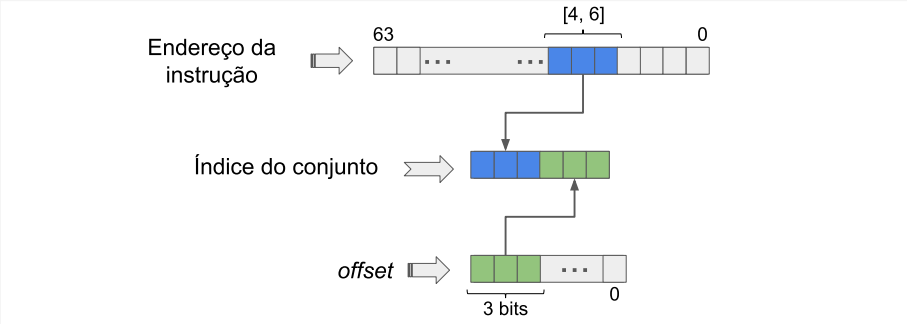


Figura 4.4: Mapeamento de um endereço de instrução e um *offset* para um índice de conjunto da AHT do DEWP.

Após a criação do índice, procura-se uma entrada do conjunto indexado contendo os 16 bits menos significativos do endereço da instrução em seu campo PC e os 3 bits mais significativos de *offset* no campo Deslocamento. Se essa entrada for encontrada, sua previsão é copiada para os

metadados da linha de cache. Caso o bit Vínculo da entrada tenha o valor 0, ele recebe o valor 1 e o campo Ponteiro do bloco passa a apontar para a entrada da AHT, como é ilustrado na Figura 4.5.

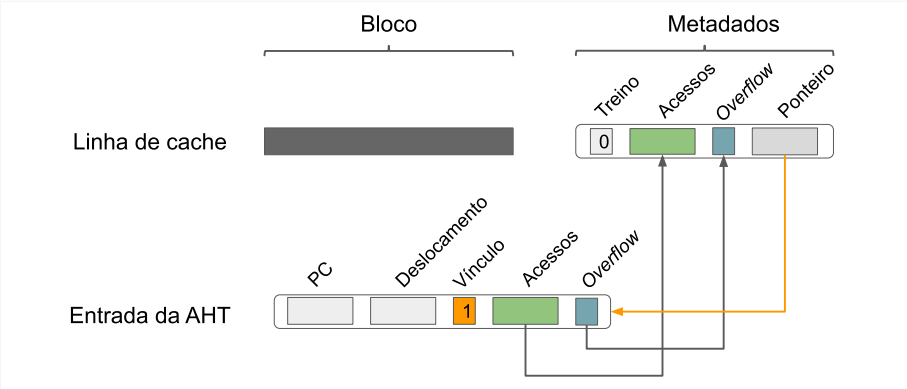


Figura 4.5: Cópia de valores de uma entrada da AHT do mecanismo DEWP para os metadados de uma linha de cache.

Se nenhuma entrada for encontrada, o mecanismo não possui uma previsão pronta para o bloco e deve ser treinado. Para isso, a entrada que recebeu seu último acesso a mais tempo (LRU) é limpa e vinculada ao bloco para seu treinamento. Esse vínculo resulta no estado ilustrado pela Figura 4.6 e ocorre seguindo o seguinte conjunto de passos:

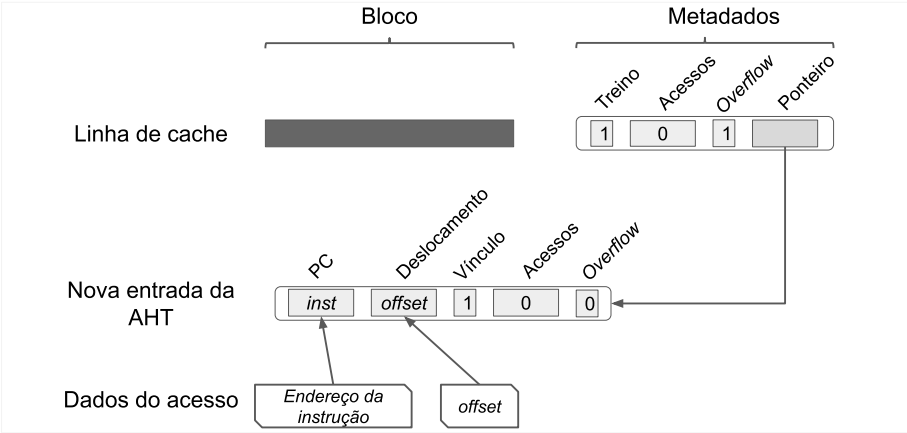


Figura 4.6: Alocação de nova entrada na AHT, com vínculo para treinamento, no mecanismo DEWP.

- Os primeiros 16 bits do endereço da instrução são armazenados no campo PC da entrada na AHT.
- Os 3 bits mais significativos de *offset* são armazenados no campo Deslocamento da entrada na AHT.
- O bit Vínculo da entrada na AHT recebe o valor 1.
- O campo Ponteiro nos metadados do bloco passa a apontar para a entrada na AHT.
- O bit Treino nos metadados do bloco recebe o valor 1.

- O bit *Overflow* nos metadados do bloco recebe o valor 1.

Com esse vínculo, o mecanismo pode treinar a nova entrada da AHT a partir dos acessos recebidos pelo bloco vinculado a ela.

O treinamento do DEWP acontece cada vez que um evento da cache ocorre com alguma linha que tenha o valor 1 em seu bit Treino. Os eventos e as ações relacionadas a eles são descritos a seguir:

- **Acesso:** O contador de acessos da entrada na AHT vinculada à linha é incrementado. Caso já possua seu valor máximo, o bit *Overflow* recebe 1.

- **Eliminação:** O bit Vínculo da entrada na AHT vinculada à linha recebe o valor 0.

Algumas linhas de cache possuem um link para entradas da AHT sem que seu bit Treino esteja ativado. Essas linhas servem para ajustar as predições do mecanismo. Quando são eliminadas, caso seu contador de acessos não esteja zerado, seu valor é reduzido do contador de acessos de sua entrada vinculada, assim corrigindo predições exageradas. Além disso, caso uma predição erre para menos e uma dessas linhas seja acessada depois de ser prevista como morta, o contador de acessos de sua entrada vinculada é incrementado e seu bit Treino é ativado. Assim, futuros acessos irão atualizar a AHT. Como ocorre durante a eliminação de linhas do treinamento, quando blocos que possuem um link para a AHT são eliminados, seu vínculo é removido.

O panorama geral de funcionamento do DEWP durante um evento em uma linha de cache é ilustrado pelo fluxograma da Figura 4.7.

Na implementação original desse mecanismo, a AHT e os metadados em cada linha de cache possuem contadores e bits de *overflow* independentes para previsões de leitura e escrita. Como nas avaliações realizadas para este trabalho essa distinção não é necessária, esses contadores e bits foram unificados em um único contador de acessos e um bit de *overflow* no número de acessos. Essa simplificação não reduz de forma significativa a precisão do mecanismo, visto que, como ilustrado pela Figura 4.8, a maior parte das linhas de cache recebe um número muito pequeno de acessos antes de sua morte, geralmente menor que o valor máximo do contador de acessos. A partir dessa implementação, esse mecanismo obtém uma média geométrica de erros em suas previsões de 17% para as aplicações do SPEC-CPU 2006.

4.2 POLÍTICAS DE OTIMIZAÇÃO

Neste trabalho, as previsões de linhas mortas foram adotadas para aplicar as seguintes técnicas de otimização do uso da cache:

- **Prioridade:** Entre a morte de uma linha de cache e sua substituição há um período durante o qual ela ocupa, desnecessariamente, recursos do sistema. Além disso, é possível que a política de substituição vigente na cache selecione linhas vivas para eliminação antes dela. Desta forma, a política de priorização na eliminação de linhas mortas, denominada apenas Prioridade no restante deste documento, consiste do aprimoramento da política de substituição adotada pela cache com o aumento da prioridade de linhas consideradas mortas para substituições. Assim, a política de substituição original da cache é adotada apenas no caso de não haver linhas consideradas mortas para substituição.

- ***Bypass:*** Como a Figura 4.8 mostra, a maior parte dos acessos recebidos pelos blocos de cache é único, ou seja, ao serem instalados nas caches estes dados já podem ser

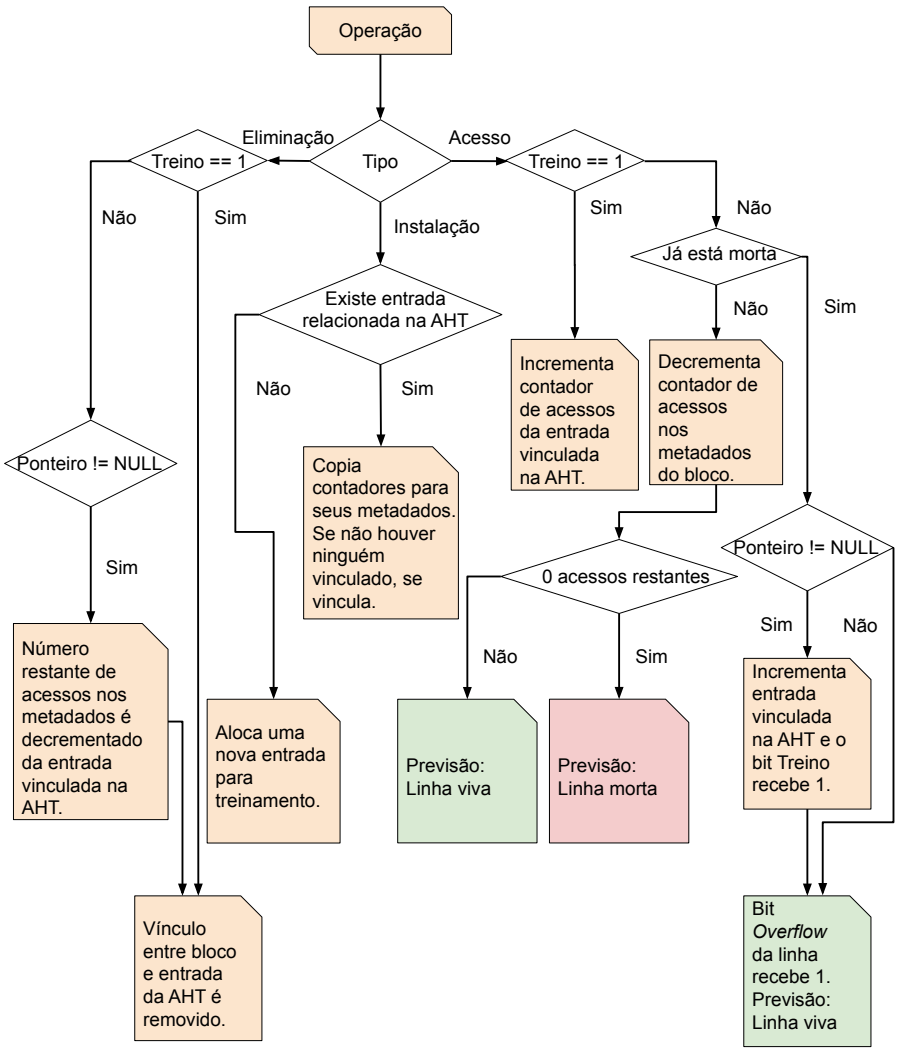


Figura 4.7: Panorama geral do funcionamento do DEWP durante um evento em um bloco da cache.

considerados mortos. A instalação desses dados remove conteúdos potencialmente vivos das caches, um fenômeno conhecido como *cache trashing*. O *Bypass* é uma política que busca prevenir esse efeito, entregando o conteúdo de linhas *dead on arrival* para seu requerente sem instalá-las na cache, permitindo assim, mais tempo para possíveis futuros acessos a dados já presentes.

Como mostrado na Figura 1.2, na maior parte do tempo, as linhas de cache encontram-se mortas. Porém, mesmo neste estado, energia estática ainda é gasta na manutenção de seus conteúdos. Para reduzir este custo energético, algumas técnicas de otimização energética foram propostas, sendo que as duas avaliadas neste trabalho são descritas a seguir:

- **Gated-Vdd:** A técnica *Gated-Vdd*, proposta por Powell et al. (2000), busca reduzir quase completamente os gastos energéticos de uma linha de cache por meio de seu "desligamento". Esse "desligamento" é realizado com a adição de um transistor após a fonte ou antes do dreno de cada bloco de cache. Enquanto este transistor permanece ativo, a linha comporta-se de forma padrão, com atrasos desprezíveis em seu tempo de acesso. Porém, ao ser desligado, a linha de cache é desativada, perdendo seu conteúdo e reduzindo de forma quase completa seus gastos com energia estática. Devido a essa

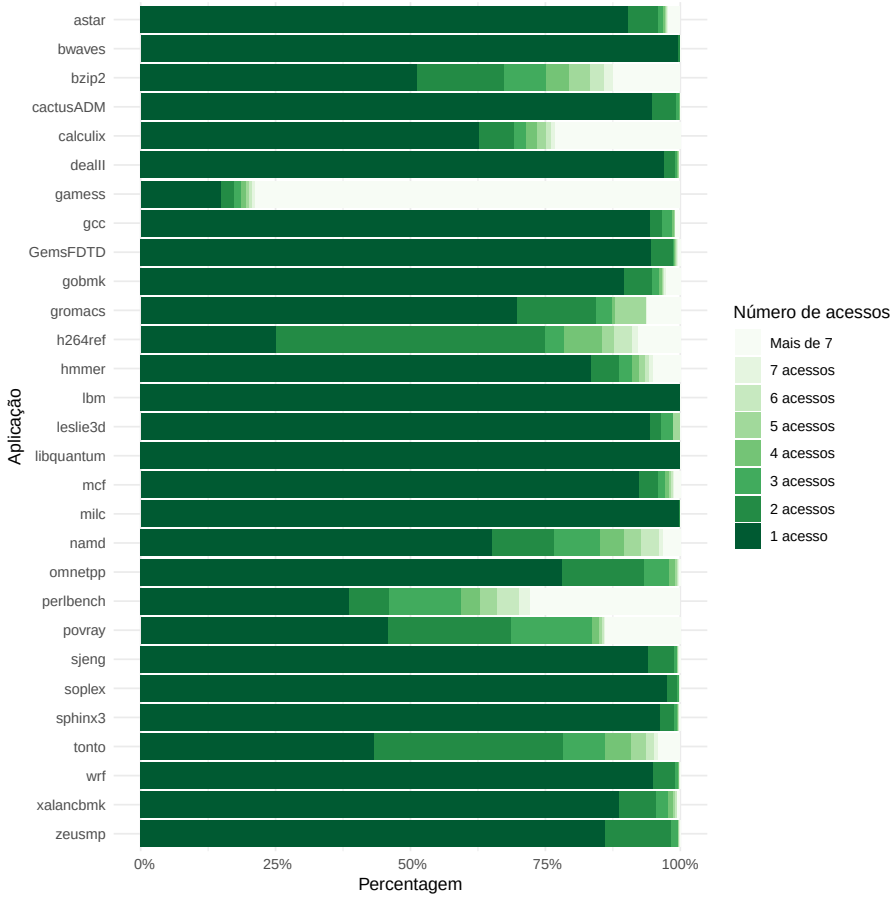


Figura 4.8: Separação de blocos da *Last Level Cache* (LLC) pelo número de acessos recebidos durante a vida.

perda de conteúdo, a aplicação do *Gated-Vdd* só pode ocorrer em linhas que não tenham sido modificadas, ou que já tenham sofrido *write-back*, de forma a manter a consistência na hierarquia de memória.

- **Drowsy:** Flautner et al. (2002) propõem um circuito capaz de alternar uma linha de cache entre o modo padrão de funcionamento e o modo *drowsy*, no qual sua tensão é reduzida. Esse corte de tensão permite que o circuito minimize seus gastos com energia estática em até 75%, porém impede a realização de leituras ou escritas na linha afetada. Essa restrição não acarreta grandes penalidades no caso de aplicações incorretas da técnica, visto que a modificação entre estados do circuito leva apenas 1 ou 2 ciclos de clock.

Essas duas técnicas de otimização energética possuem vantagens e desvantagens complementares. A maior economia da *Gated-Vdd* é eficiente para aplicações com linhas de cache mortas por grandes períodos de tempo. Já a pequena penalidade para predições incorretas da *drowsy* é útil em situações onde existe um grande número de predições incorretas.

5 AVALIAÇÃO DE TÉCNICAS ENERGÉTICAS

Diversos mecanismos preditores de linhas mortas de cache utilizam técnicas de otimização como o *Bypass*, porém poucos aplicam técnicas de redução energética. Destes, grande parte aplica a técnica *Gated-Vdd*, desligando as linhas de cache dadas como mortas, o que nem sempre é vantajoso, visto que preditores de reúso não são completamente precisos. A figura 5.1 mostra a acurácia dos preditores *Dead Line and Early Write-Back Predictor* (DEWP) e *Skewed Dead Block Predictor* (SDP) sobre cada aplicação do SPEC-CPU 2006, considerando as especificações da hierarquia de memória descritas na Tabela 5.1. Foi observado que as médias geométricas das predições incorretas realizadas pelo DEWP e pelo SDP estão em torno de 17% e 4% do número total de predições, respectivamente. Caso estas predições orientem um destes mecanismos a utilizar a técnica Gated-Vdd para desligar incorretamente uma linha de cache, o dado que deveria estar armazenado nesta linha estará indisponível durante seu próximo acesso, causando um acesso extra à memória principal.

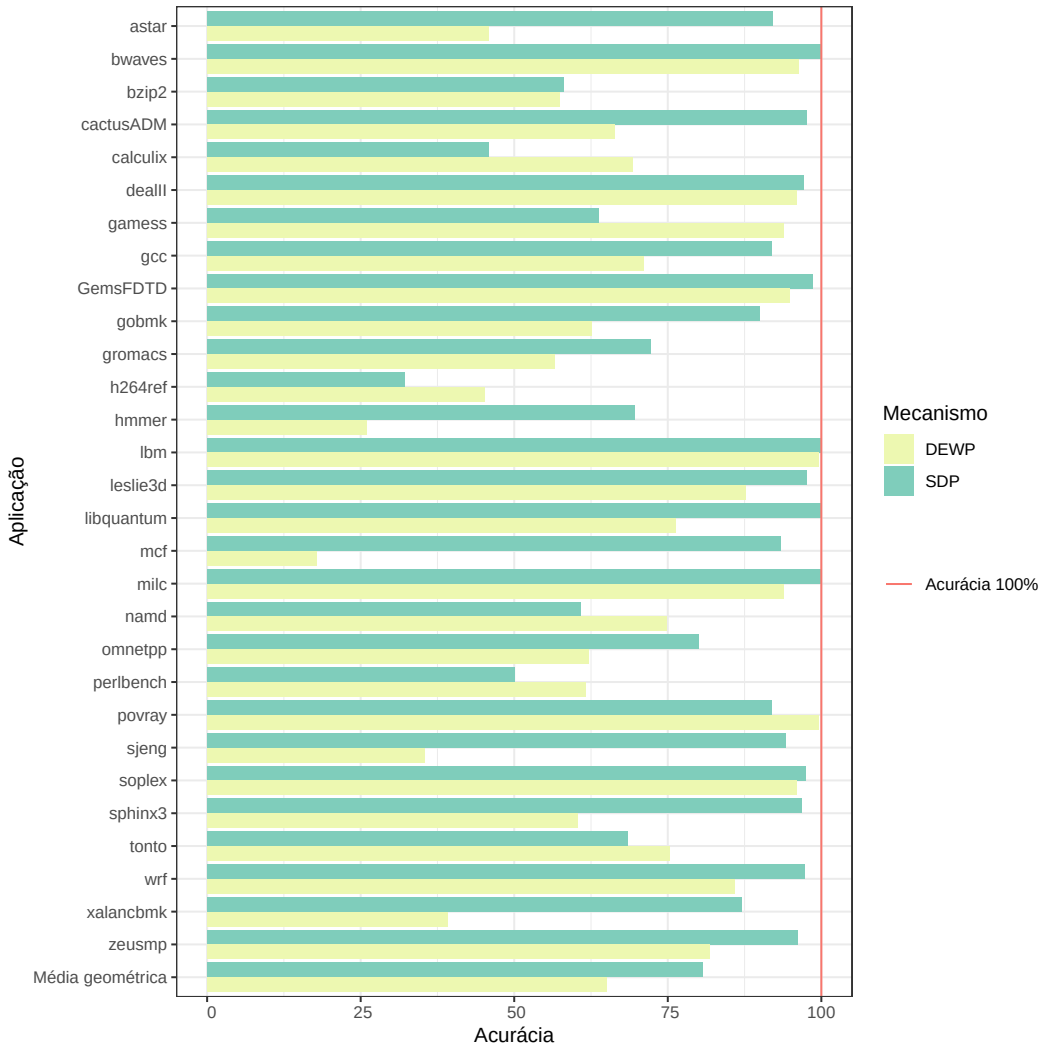


Figura 5.1: Acurácia dos mecanismos DEWP e SDP.

Além do *Gated-Vdd*, outra técnica possível de redução energética é a técnica *Drowsy*. Esta técnica, ao contrário da anterior, não apresenta a desvantagem dos acessos extras à *Dynamic*

Tabela 5.1: Configurações de memória para os experimentos do capítulo 5.

Nome	Tamanho	Associatividade	Tempo de acesso (ciclos)	Energia estática (média)	Energia dinâmica (média)
L1	32~KB	8 Ways	4	–	–
L2	256~KB	4 Ways	8	–	–
L3	2~MB	16 Ways	26	263 mW	0.198 nJ
RAM	8~GB	–	250	372 mW	3.270 nJ

Random Access Memory (DRAM), visto que as linhas nas quais é aplicada não perdem seu conteúdo. Apesar disso, nenhum dos trabalhos relacionados consultados a utiliza. Assim, esta seção dedica-se a comparar os possíveis ganhos desta técnica com o clássico *Gated-Vdd*, buscando compreender os prós e contras de cada um. Para esta análise, as duas técnicas foram implementadas sobre os dois preditores descritos no capítulo 4, para assim demonstrar a independência entre a técnica adotada e o preditor utilizado.

Os mecanismos e técnicas utilizados nesta seção foram implementados sobre a *Last Level Cache* (LLC) considerando a hierarquia de memória descrita na Tabela 5.1, baseada na hierarquia de memória utilizada pelo processador Core i7-4770 com a arquitetura Haswell da Intel (Raviraj, 2018). Como nesse processador o último nível de cache de 8MB é compartilhado entre 4 núcleos, foi considerado que apenas um quarto desse espaço (2MB) pôde ser utilizado pelas aplicações.

As próximas seções apresentam análises a respeito das duas técnicas, detalhando os resultados obtidos e suas consequências, sempre avaliados em relação a uma LLC sem nenhum preditor de reúso implementado (*baseline*).

5.1 ACESSOS À DRAM

A vantagem da técnica *Drowsy* sobre o *Gated-Vdd* está na eliminação dos acessos adicionais à DRAM causados pelo desligamento incorreto de blocos da cache. A figura 5.2 mostra a percentagem adicional de misses causados pelo uso do *Gated-Vdd* com os mecanismos DEWP e SDP. Considerando a média aritmética, este efeito colateral acarreta um aumento de 11% no número de acessos à DRAM. Porém, para algumas aplicações como *gamess* e *h264ref*, esta percentagem ultrapassa os 20%, chegando a mais de 80% para a aplicação *calculix* utilizando o SDP, indicando grandes possibilidades de economia, tanto energéticas quanto temporais.

5.2 TEMPO DE EXECUÇÃO

Como apontado na seção anterior, a técnica *Drowsy* apresenta como grande vantagem sua redução completa do número de misses adicionais causados à DRAM pelo *Gated-Vdd*. Tendo em vista que o tempo de acesso à DRAM é em torno de uma ordem de grandeza maior que à LLC, esta redução dos acessos apresenta grande potencial para ganhos de desempenho na execução de aplicações. A Figura 5.3 mostra, em relação ao *baseline*, a percentagem de tempo de execução adicional gasto por cada técnica sobre os mecanismos avaliados. A primeira linha representa os gastos adicionais referentes ao mecanismo DEWP, enquanto a segunda ao SDP. Para cada um dos mecanismos, a primeira coluna apresenta sua perda de desempenho com a utilização do *Gated-Vdd* e a segunda ao ser aplicada a técnica *Drowsy* .

Os resultados apresentam, para ambos os preditores, o mesmo padrão de atrasos entre as duas técnicas implementadas, em decorrência da independência entre acurácia do preditor e

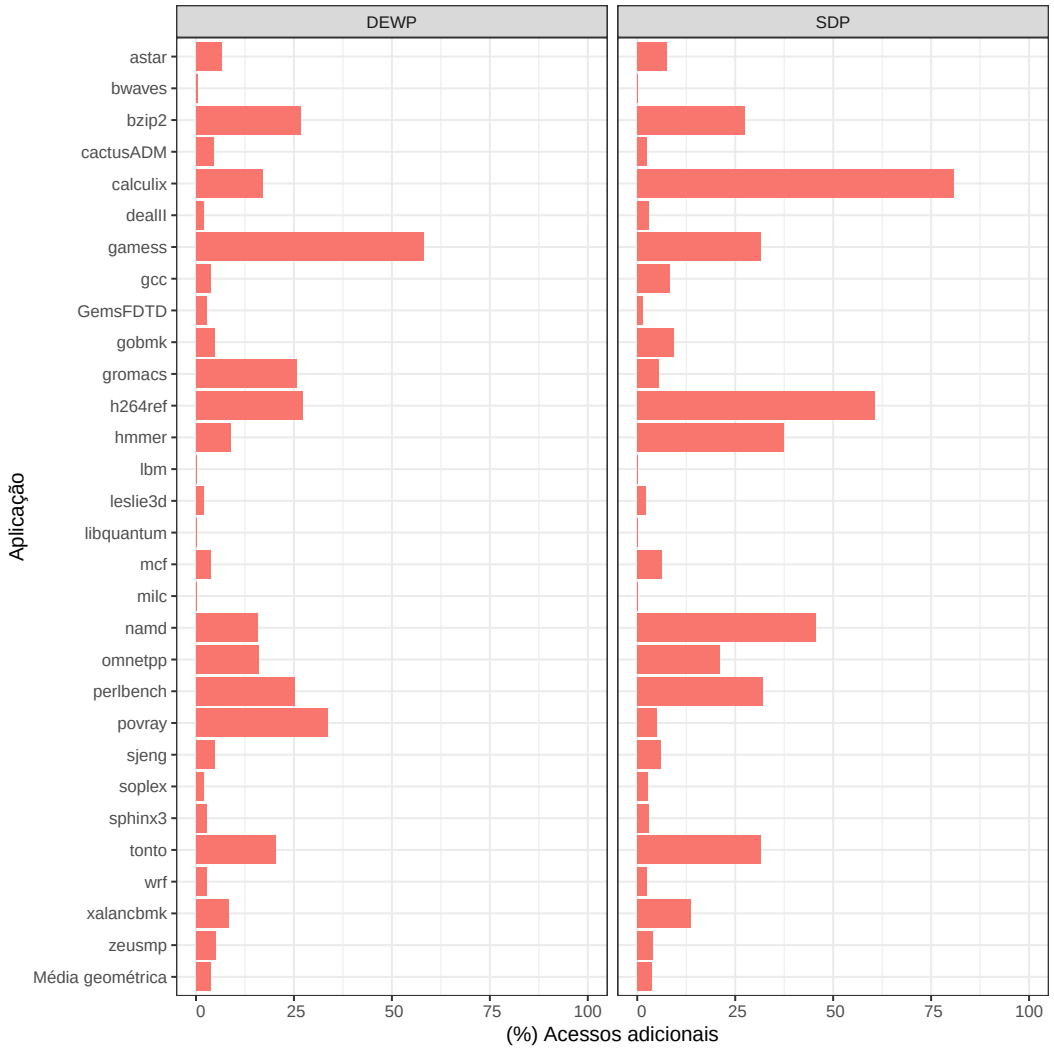


Figura 5.2: Acessos adicionais à DRAM causados pelo desligamento incorreto de linhas de cache com a técnica *Gated-Vdd* orientada pelos preditores DEWP e SDP em relação ao *baseline*, um sistema sem nenhum preditor de linhas mortas e técnica de otimização. O *baseline* é representado pelo 0%, barras maiores indicam um maior aumento no número de acessos à DRAM. Assim, uma barra alcançando 10% significa que a combinação de mecanismo e técnica que a gerou possui 10% mais acessos à DRAM que o *baseline* durante a execução da mesma aplicação.

sua técnica de redução energética. Apesar deste padrão ser mantido, as aplicações sob influência da técnica *Drowsy* apresentam um percentual de atraso médio duas ordens de grandeza menor que aquele obtido aplicando-se o *Gated-Vdd*. Isso porque cada erro de predição que gera um acesso extra à DRAM com o *Gated-Vdd* acarreta apenas uma alteração de tensão com o *Drowsy*, levando a somente 1 ou 2 ciclos de atraso. Essa redução afeta não só o tempo de execução das aplicações mas também a energia gasta pelo sistema como um todo.

Apesar disso, o *Gated-Vdd* apresenta vantagens quanto à redução de energia estática da LLC, o que é explorado na seção seguinte.

5.3 ENERGIA ESTÁTICA DA LLC

O objetivo das técnicas de redução energética é minimizar os gastos com energia estática de partes de uma cache. Como mostrado nas seções anteriores, a técnica *Drowsy* apresenta grandes vantagens em relação ao desempenho quando comparada ao *Gated-Vdd*. Porém, em um

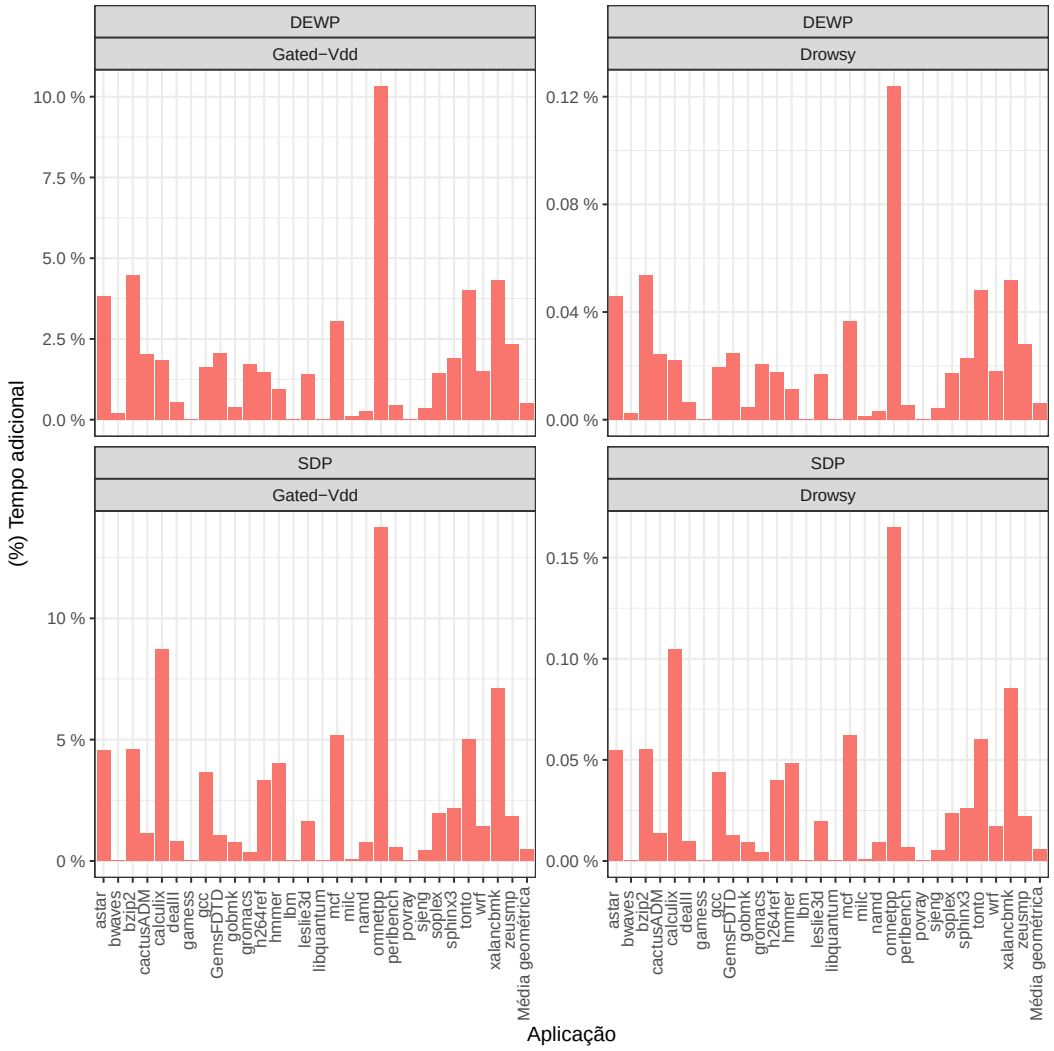


Figura 5.3: Tempo de execução adicional gerado pela aplicação das técnicas *Gated-Vdd* e *Drowsy* orientadas pelos preditores DEWP e SDP em relação ao *baseline*, um sistema sem nenhum preditor de linhas mortas e técnica de otimização. O *baseline* é representado pelo 0%, colunas maiores indicam tempos maiores de execução. Assim, uma coluna alcançando 10% significa que a combinação de mecanismo e técnica que a gerou possui um tempo de execução 10% maior que o do *baseline* durante a execução da mesma aplicação.

mecanismo preditor ideal, a técnica *Gated-Vdd* está em vantagem ao conseguir economizar quase totalmente a energia estática de uma linha, enquanto com o *Drowsy* em torno de 25% da energia continua sendo gasta. A Figura 5.4 mostra a percentagem de energia estática economizada em relação ao *baseline* por cada mecanismo analisado com ambas as técnicas de economia de energia. Na primeira linha temos as avaliações sobre o mecanismo DEWP quanto ao uso das técnicas *Gated-Vdd* e *Drowsy* e na segunda as mesmas avaliações quanto ao SDP. Apesar dos padrões de economia serem mantidos para ambas as técnicas, o *Gated-Vdd* aumenta em 10% a economia de energia quando comparado aos mesmos mecanismos com o *Drowsy* em relação ao *baseline*. Porém, mesmo com uma maior economia, em aplicações como *gamess* e *h264ref*, ganhos inferiores a 10% foram obtidos. Esses pequenos ganhos podem ser explicados pela maior reutilização de dados nestas aplicações, como mostrado na Figura 4.8. A maior reutilização leva a menores períodos de morte e, consequentemente, menos oportunidades para otimização. Ainda assim, os mecanismos implementando a técnica *Gated-Vdd* obtiveram uma média geométrica de ganhos de aproximadamente 50% em relação ao *baseline*, demonstrando seu potencial e

superioridade em relação à economia energética quando comparado ao *Drowsy*, com uma média geométrica próxima a 40%. Além da energia estática da LLC, outros tipos de energia afetam e são afetados pelos preditores de linhas mortas e serão explorados na próxima seção.

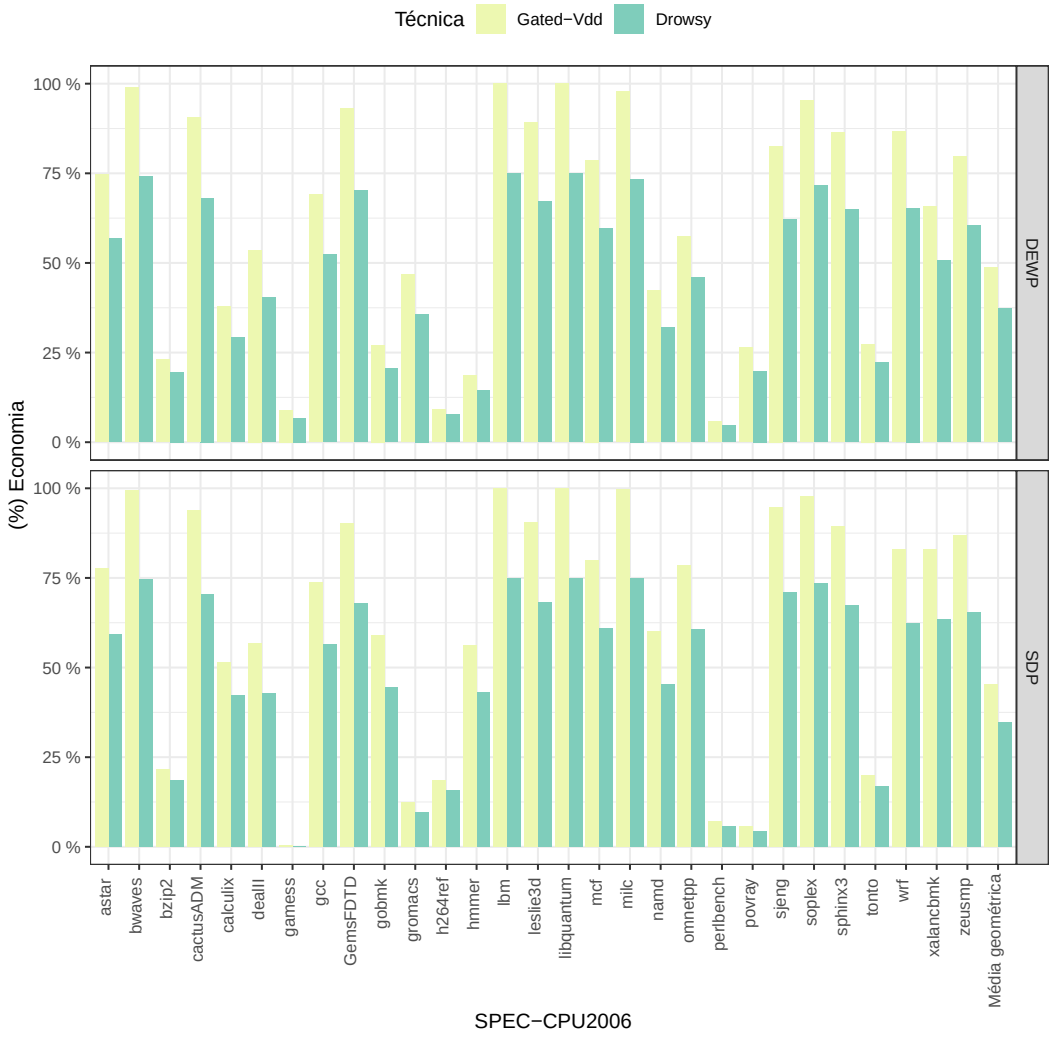


Figura 5.4: Economia de energia estática na LLC das técnicas *Gated-Vdd* e *Drowsy* orientadas pelos preditores DEWP e SDP em relação ao *baseline*, um sistema sem nenhum preditor de linhas mortas e técnica de otimização. O *baseline* é representado pelo 0%, colunas maiores indicam consumos energéticos menores. Assim, uma coluna alcançando 10% significa que a combinação de mecanismo e técnica que a gerou consome apenas 90% da energia consumida pelo *baseline* durante a execução da aplicação.

5.4 ENERGIA TOTAL

As técnicas de economia energética aplicadas por preditores de linhas mortas, neste trabalho, tentam reduzir os gastos com energia estática diretamente na LLC. Porém, diversas outras fontes de gastos existem em um sistema. Esta seção trata desses gastos, considerando as duas estruturas mais diretamente afetadas pelas técnicas *Gated-Vdd* e *Drowsy*, a LLC e a DRAM. Na DRAM serão considerados seus gastos estáticos e dinâmicos, enquanto na LLC apenas os gastos estáticos serão considerados, visto que seu número de acessos não é alterado pelas técnicas analisadas. A Figura 5.5 mostra a energia economizada pelos diversos preditores e técnicas em relação ao *baseline*. Diferentemente dos gráficos relacionados ao tempo e à energia estática, neste

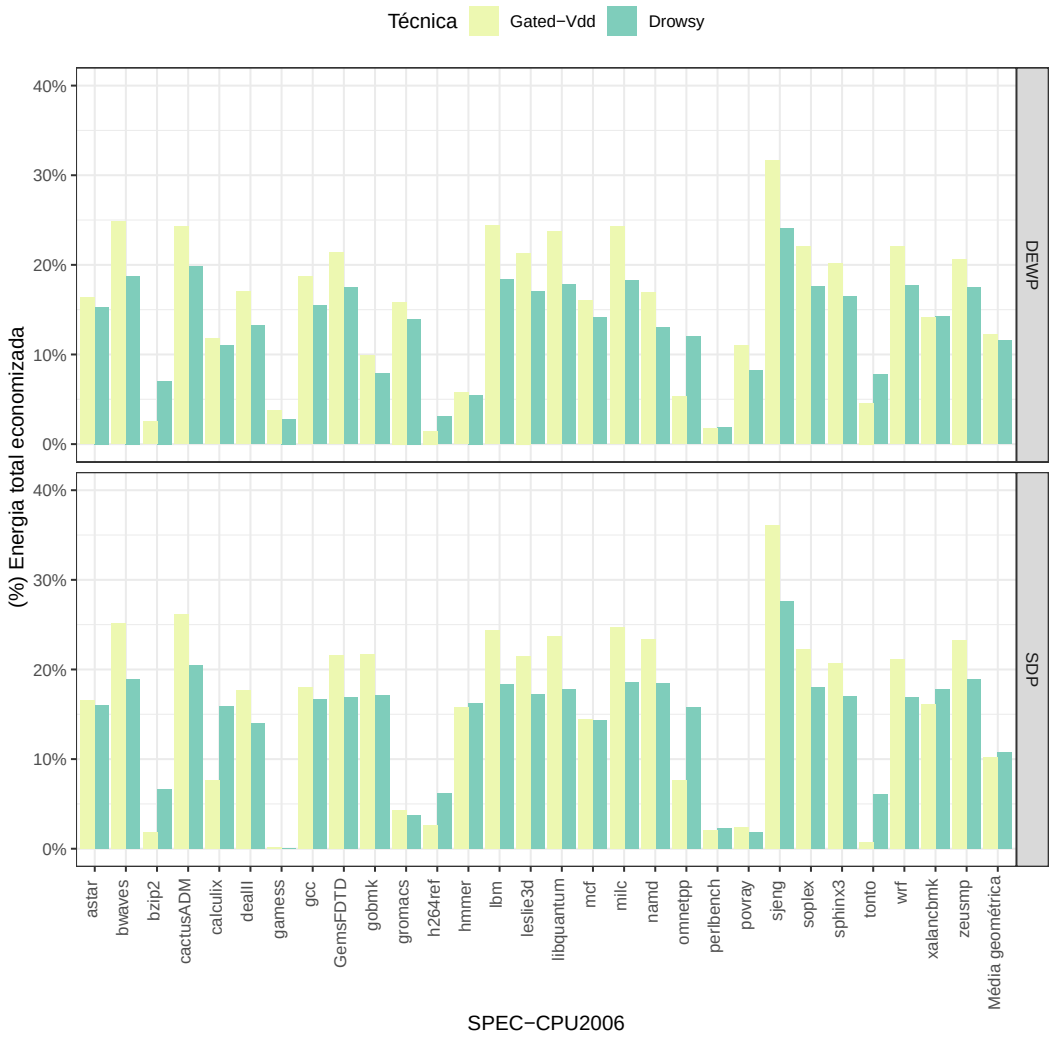


Figura 5.5: Economia na energia total (LLC + DRAM) das técnicas *Gated-Vdd* e *Drowsy* orientadas pelos preditores DEWP e SDP em relação ao *baseline*, um sistema sem nenhum preditor de linhas mortas e técnica de otimização. O *baseline* é representado pelo 0%, colunas maiores indicam consumos energéticos menores. Assim, uma coluna alcançando 10% significa que a combinação de mecanismo e técnica que a gerou consome apenas 90% da energia consumida pelo *baseline* durante a execução da aplicação.

não há grande variação entre os mecanismos, apenas uma diferença de aproximadamente 3% entre as versões com *Gated-Vdd* e *Drowsy*, sendo que a primeira técnica leva vantagem. Além disso, todos os mecanismos foram capazes de economizar energia, sendo que a média aritmética de economia com o *Gated-Vdd* foi em torno de 16% enquanto com o *Drowsy* 14%. Para analisar de onde vem esta economia adicional do *Gated-Vdd*, consideremos o mecanismo DEWP.

A Figura 5.6 mostra os gastos energéticos da implementação do DEWP com a técnica *Gated-Vdd* em relação ao mesmo mecanismo usando a técnica *Drowsy*. Na primeira linha temos comparações em relação à DRAM. Na segunda temos a energia estática da LLC à esquerda e a energia total do sistema à direita. Colunas contendo valores acima de 100% resultam de experimentos onde a utilização da técnica *Gated-Vdd* consumiu mais energia que o uso da *Drowsy*. Já colunas com valores mais baixos indicam um maior consumo energético com o uso da técnica *Drowsy* em relação à utilização do *Gated-Vdd*.

A primeira linha é dominada pela técnica *Drowsy* porque, ao causar mais acessos à DRAM, o *Gated-Vdd* aumenta seus gastos com energia dinâmica. Além disso, estes acessos

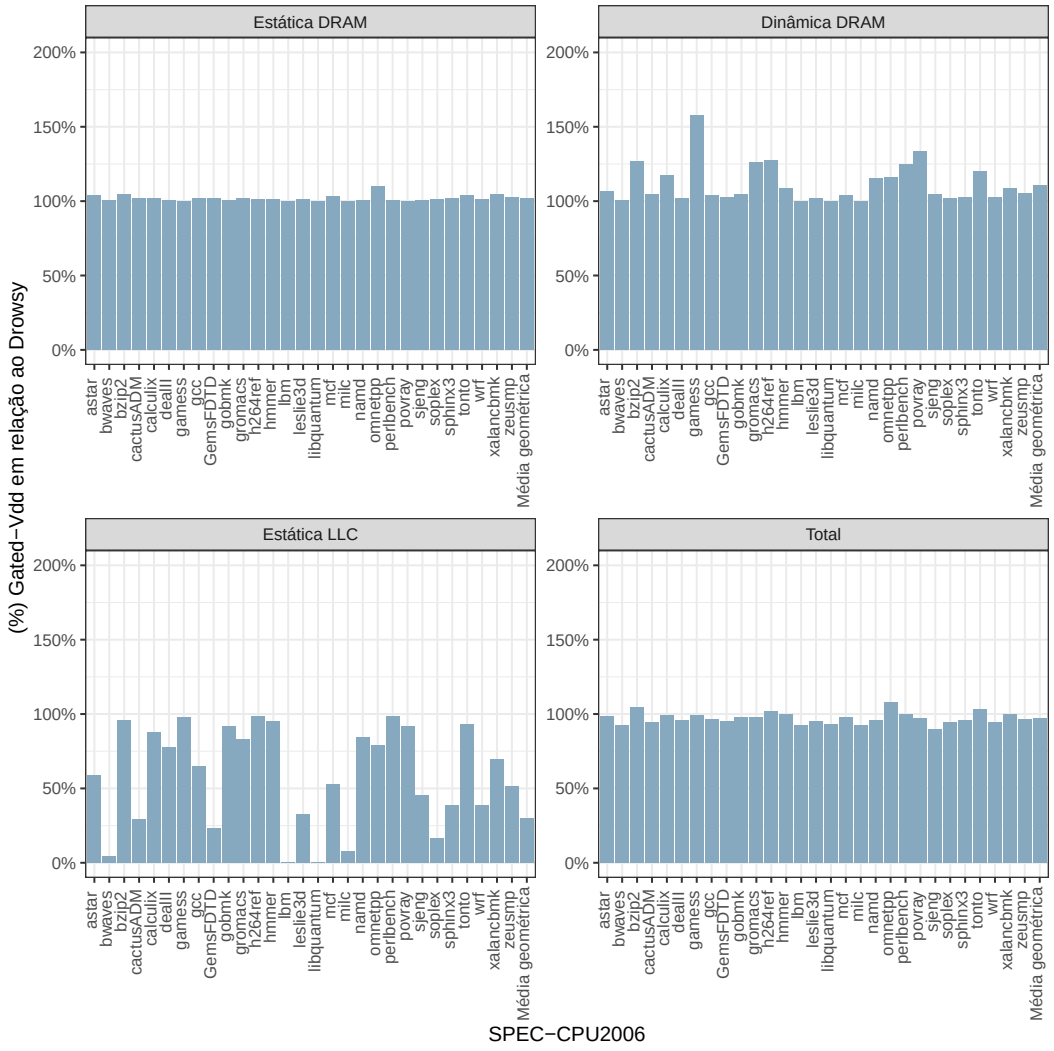


Figura 5.6: Comparação de diversos tipos de energia consumidos pela implementação do DEWP com *Gated-Vdd* em relação ao mesmo mecanismo aplicando a técnica *Drowsy*. A aplicação da técnica *Drowsy* representa o 100% de cada gráfico. Colunas maiores que esse limite indicam consumos energéticos maiores com a técnica *Gated-Vdd*, enquanto colunas mais baixas indicam economias energéticas com essa técnica.

adicionais aumentam o tempo de execução dos programas, como mostrado na Figura 5.3, causando maiores gastos de energia estática pelos circuitos. Já o gráfico referente aos gastos estáticos da LLC mostra uma grande recuperação da técnica *Gated-Vdd*. Nesse gráfico, os ganhos da técnica *Gated-Vdd* sobre a *Drowsy* alcançam uma média geométrica de 20%. Isso é causado pela maior capacidade de redução de gastos estáticos do *Gated-Vdd*, o que resulta em uma grande vantagem. Essa vantagem nos gastos estáticos da LLC equilibra o resultado final da energia total, com um ganho médio de 3% para a técnica *Gated-Vdd*.

5.5 CONCLUSÃO

Conforme novas fontes de gastos energéticos do sistema são consideradas, o tempo de execução torna-se um fator crítico em sua economia. A técnica *Drowsy* apresenta grandes vantagens em relação ao *Gated-Vdd*, considerando o desempenho geral do sistema devido à sua redução no tempo de execução das aplicações. Porém, estes ganhos em tempo de execução também podem ser obtidos utilizando-se técnicas de otimização, como o *Bypass*. O próximo

capítulo apresenta uma avaliação sobre duas técnicas de otimização de desempenho, a Prioridade e o *Bypass*, quando aplicadas pelos preditores DEWP e SDP, em conjunto com a técnica de redução energética mais utilizada, *Gated-Vdd*, para assim buscar compreender suas influências sobre a LLC.

6 UNIÃO DE POLÍTICAS

As técnicas de redução energética estudadas no capítulo anterior não são as únicas políticas adotadas por preditores de linhas mortas a partir de suas previsões. Duas técnicas de otimização bastante utilizadas por estes preditores são a Prioridade e o *Bypass*. Como explicado na seção 4.2, a política de Prioridade aumenta a preferência da política de substituição de linhas por linhas dadas como mortas. Já o *Bypass* evita que linhas *dead on arrival*, ou seja, linhas que receberão apenas um acesso durante toda sua vida em cache, sejam instaladas. A aplicação destas políticas altera o comportamento da cache, permitindo que ela seja mais eficiente. Porém, da mesma forma que a técnica *Gated-Vdd*, sua utilização incorreta afeta negativamente o desempenho do sistema. Além disso, como a dinâmica da cache é modificada, estas técnicas se influenciam mutuamente, podendo ter resultados inusitados. Assim, o objetivo deste capítulo é estudar a influência mútua destas políticas através dos dois preditores de linhas mortas utilizados no capítulo anterior, *Dead Line and Early Write-Back Predictor* (DEWP) e *Skewed Dead Block Predictor* (SDP), aplicados sobre a *Last Level Cache* (LLC). Nos experimentos deste capítulo, todos os preditores estarão utilizando a técnica *Gated-Vdd*, visto que, é a mais utilizada pelos preditores de linhas mortas. As configurações de hierarquia de memória utilizadas são mostradas na Tabela 6.1. As alterações na configuração da *Dynamic Random Access Memory* (DRAM) simulam melhor o comportamento da execução de uma aplicação em um ambiente não controlado, visto que nem toda a DRAM é utilizada por um único processo. Já as caches tiveram suas especificações atualizadas para simular arquiteturas mais recentes da intel que têm maior capacidade de armazenamento.

Tabela 6.1: Configurações de memória para os experimentos do capítulo 6.

Nome	Tamanho	Associatividade	Tempo de acesso (ciclos)	Energia estática (média)	Energia dinâmica (média)
L1	32~KB	8 Ways	4	–	–
L2	256~KB	8 Ways	8	–	–
L3	4~MB	16 Ways	26	309 mW	0.294 nJ
RAM	1~GB	–	200	8 mW	3.760 nJ

6.1 PRIORIDADE E BYPASS

Técnicas de otimização como Prioridade e *Bypass* alteram o comportamento padrão da cache de forma a otimizar seu desempenho. As alterações causadas por estas técnicas mudam não somente o efeito de outras otimizações, como também as próximas predições realizadas pelos mecanismos preditores de linhas mortas, podendo causar efeitos indesejados nas caches. Esta seção tem por objetivo estudar o efeito destas técnicas no desempenho dos mecanismos de predição de reúso, DEWP e SDP, analisando ganhos e perdas energéticos e temporais causados por cada técnica. Para este estudo, a política de redução energética *Gated-Vdd* foi implementada em cada preditor, sendo ativada no momento previsto para a morte de cada linha. Já as políticas de otimização Prioridade e *Bypass* foram implementadas ou não, dependendo do experimento, de forma a analisar seu efeito em cada mecanismo. Nos casos em que estas técnicas foram implementadas, sua ativação ocorre no instante em que uma linha é dada como morta, sem alterações entre os limiares da Prioridade, do *Bypass* e do desligamento das linhas para o SDP.

Isto acontece para prover certo grau de isonomia perante o DEWP, no qual as técnicas são aplicadas quando seus contadores chegam a 0.

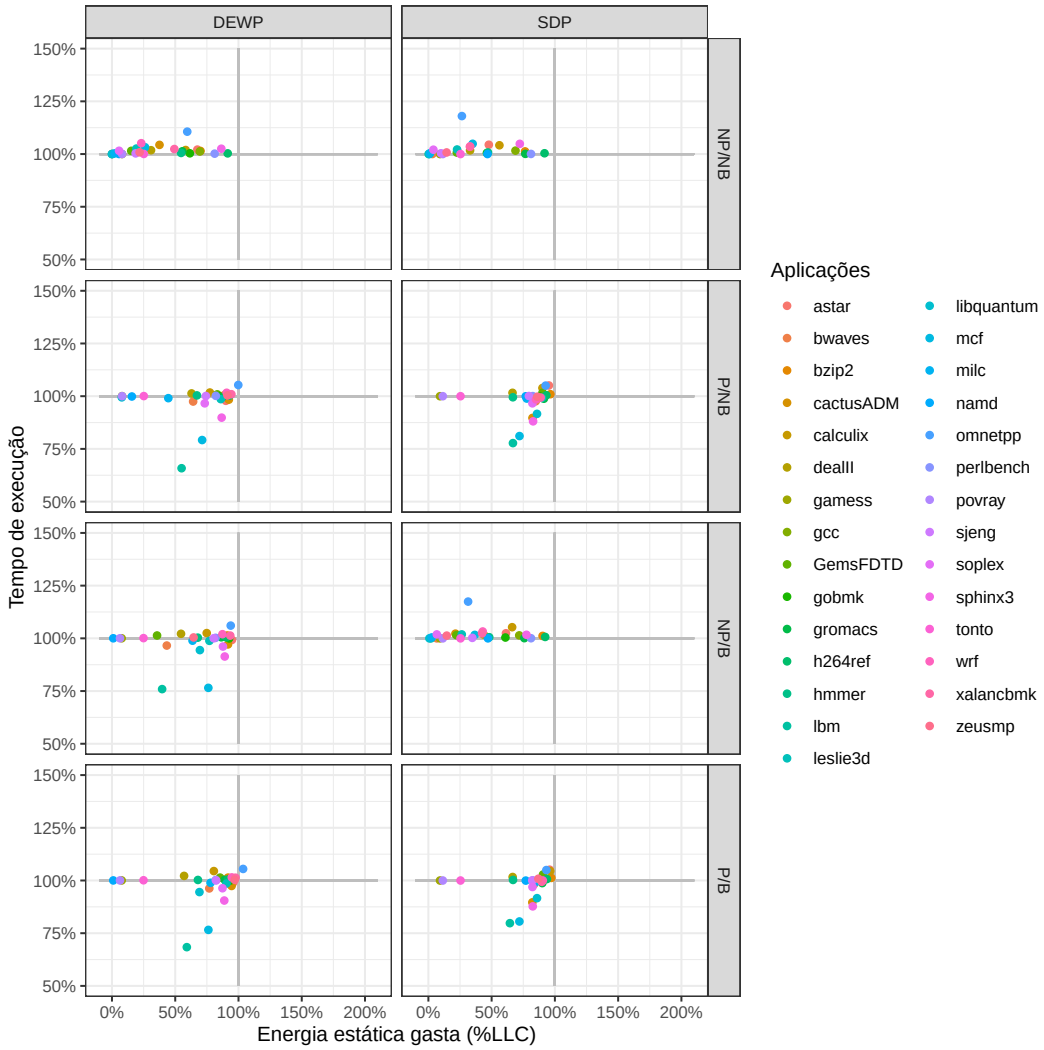


Figura 6.1: Consumo energético e tempo de execução de cada combinação de mecanismo e técnicas sobre o baseline. Todas as combinações implementam o *Gated-Vdd*, experimentos implementando a Prioridade são indicados por "P", enquanto os que não a utilizam são indicados por "NP". Já os que implementam a *Bypass* são indicados por "B", enquanto os que não a utilizam são indicados por "NB". Cada coluna representa predições orientadas por um mecanismo preditor de linhas mortas distinto (DEWP ou SDP).

A Figura 6.1 mostra o consumo energético e o tempo de execução de cada aplicação do SPEC-CPU 2006 para cada variação de mecanismo testado, com relação ao baseline. Cada coluna indica a utilização de um mecanismo: à esquerda as variações implementadas sobre o DEWP, e à direita as mesmas análises realizadas com o SDP. Para cada mecanismo, as linhas indicam a variação na utilização das técnicas Prioridade e *Bypass*. Variações com a técnica de Prioridade são indicadas por “P”, enquanto as que não a utilizam são indicadas por “NP”. Para o *Bypass*, a mesma convenção é aplicada; linhas com experimentos que o aplicam são indicadas por “B”, enquanto as que não o adotam são indicadas por “NB”. Em cada um dos gráficos, o eixo horizontal representa a energia estática consumida pela LLC e o eixo vertical refere-se ao tempo de execução de cada aplicação, ambos relativos ao baseline, ou seja, caso nem o tempo de execução nem os gastos energéticos sejam alterados para determinada aplicação, essa deve estar posicionada ao centro do gráfico, na posição (100%, 100%). Porém, caso uma

aplicação apresente uma economia energética de 10% e um aumento no tempo de execução de 5%, em relação ao baseline, será posicionada em (90%, 105%). É direto que praticamente todos os mecanismos/técnicas conseguem reduzir a energia gasta pela LLC. A aplicação *omnetpp*, durante a utilização de ambas as técnicas sobre o DEWP, é a única a consumir mais energia que o baseline. Isso ocorre porque, além de seu tempo de execução aumentar, as técnicas de otimização aplicadas reduzem suas possibilidades de economia pelo desligamento de linhas mortas. O aumento no tempo de execução dessa aplicação acontece em grande parte dos experimentos e é causado pelo desligamento incorreto de seus blocos (como é indicado pelos gráficos da primeira linha). Porém, conforme são adicionadas técnicas de otimização, este efeito negativo no tempo de execução é reduzido. A primeira linha de experimentos, sem a adoção de nenhuma técnica de otimização, mostra o efeito esperado da aplicação do *Gated-Vdd*, com grandes economias energéticas e degradações nos tempos de execução ocasionadas pelas predições e consequentes desligamentos incorretos.

Já a segunda e a quarta linhas apresentam dinâmicas semelhantes, com a redução no consumo energético para quase todas as aplicações e redução no tempo de execução de, em média, 40% delas, indicando que, após a adoção da Prioridade, a adição do *Bypass* não fornece vantagens, podendo até mesmo contribuir para a deterioração no desempenho de algumas aplicações. Essa inibição é decorrente da redução na eliminação prematura de linhas vivas pela priorização das linhas mortas durante misses, reduzindo o cache trashing (substituição de linhas vivas por *dead-on-arrival*) e simulando as vantagens do *Bypass*.

A terceira linha, com resultados implementando o *Bypass* sem Prioridade, apresenta resultados divergentes quando considerados ambos os mecanismos. Durante a utilização do DEWP, esta técnica sozinha conseguiu otimizar o tempo de execução de tantas aplicações quanto o experimento com ambas as técnicas de otimização, mantendo baixo consumo energético. Já ao ser aplicada sobre o SDP, gerou resultados semelhantes ao experimento sem nenhuma política de otimização, indicando que praticamente não é utilizada por este mecanismo.

Diversos experimentos obtiveram resultados parecidos, então, a fim de refinar as conclusões obtidas, a Figura 6.2 mostra as médias geométricas dos desempenhos das aplicações utilizadas no experimento anterior. Cada coluna indica qual mecanismo preditor foi utilizado. Cada combinação de técnicas é representada por uma cor, e está posicionada com relação à média do consumo de energia pela LLC na horizontal e à média do tempo de execução na vertical, ambos relativos ao baseline. Esta figura evidencia ainda mais as conclusões previamente obtidas. Com relação ao DEWP, a única combinação a perder desempenho é aquela sem nenhuma técnica de otimização, visto que sua única interferência com relação ao tempo de execução é causada por misses de linhas incorretamente desligadas. Com relação às outras combinações, a utilização da priorização de linhas mortas anulou completamente os efeitos positivos do *Bypass*, apesar deste sozinho atingir ganhos próximos à Prioridade. Com relação à energia, todas as variações que obtiveram ganhos de desempenho apresentaram ganhos energéticos similares. Já o mecanismo sem nenhuma técnica de otimização conseguiu uma economia média 40% maior que os outros, isso porque ao não aplicar nenhuma técnica de otimização, mantém suas linhas de cache desligadas por mais tempo, gerando maiores ganhos. Os experimentos realizados com o SDP apresentam resultados similares aos do DEWP, com exceção do mecanismo aplicando apenas a técnica *Bypass*, que apresenta resultados muito próximos ao experimento apenas contendo o *Gated-Vdd*, o que comprova a conclusão do gráfico anterior, além de mostrar que nem toda combinação de técnicas pode ser adequada para determinado mecanismo, devendo esta combinação ser escolhida cautelosamente.

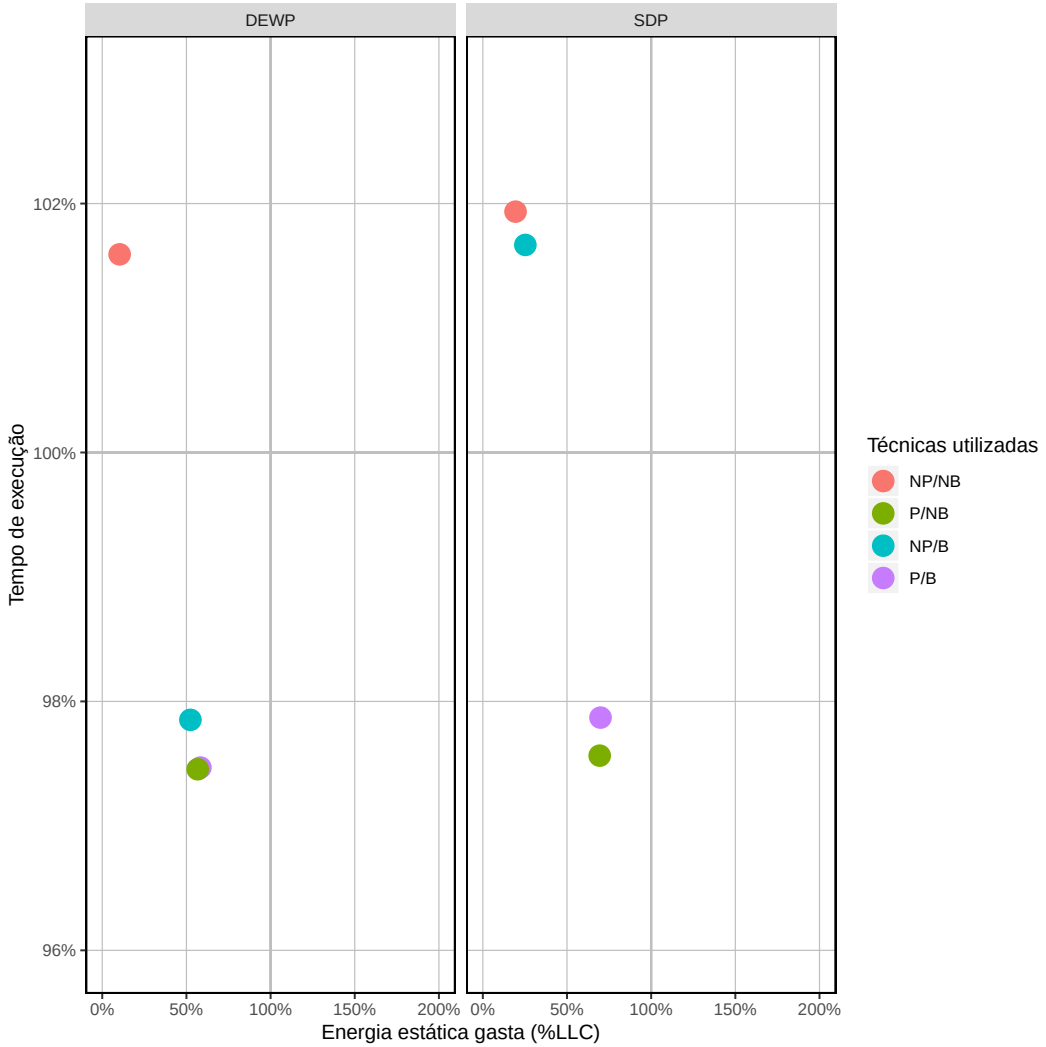


Figura 6.2: Média geométrica do tempo de execução e do consumo energético de todas as aplicações do SPEC-CPU 2006 para cada combinação de mecanismo e técnicas sobre o baseline. Todas as combinações implementam o *Gated-Vdd*, experimentos implementando a Prioridade são indicados por "P", enquanto os que não a utilizam são indicados por "NP". Já os que implementam a *Bypass* são indicados por "B", enquanto os que não a utilizam são indicados por "NB". Cada coluna representa predições orientadas por um mecanismo preditor de linhas mortas distinto (DEWP ou SDP).

6.2 CONCLUSÃO

Neste capítulo foram abordadas as consequências da junção de duas conhecidas técnicas de otimização de políticas de substituição, Prioridade e *Bypass*, com a política de redução energética *Gated-Vdd*, sobre dois preditores de linhas mortas de cache, os mecanismos DEWP e SDP. A partir desta análise foi possível concluir que, para ambos os preditores, a adoção da política Prioridade gerou uma redução de 2% na média geométrica do tempo de execução das aplicações, a qual foi mantida ou degradada com a adoção do *Bypass*. Já ao adotar o *Bypass* em conjunto com o *Gated-Vdd*, apenas o mecanismo DEWP obteve uma melhora no desempenho de suas execuções, indicando que nem sempre a adoção de uma nova politica de otimização fornecerá ganhos, devendo a influência mútua de cada política em um dado preditor ser estudada antes de sua utilização. Quanto à energia, todas as combinações de técnicas foram capazes de otimizar os gastos de energia estática na LLC, com os principais ganhos vindo dos experimentos aplicando apenas o *Gated-Vdd*. O próximo capítulo contém a conclusão deste trabalho, abordando

a consequência destes resultados em relação ao crescente aumento da relevância da energia estática nos circuitos atuais.

7 CONCLUSÕES

Reduções no tamanho dos transistores têm permitido o crescimento da concentração deste componente nos chips atuais. A fim de manter o desempenho destes circuitos com complexidades crescentes, constantemente são realizadas reduções em suas tensões, minimizando também seus gastos com energia dinâmica. Conforme essa tensão decai e a energia dinâmica perde relevância, a energia estática aparece como um novo problema.

As caches, devido ao seu tamanho, apresentam um consumo elevado de energia estática, sendo que parte significativa desta energia é gasta na manutenção de conteúdos mortos. Uma solução para este problema são os chamados preditores de linhas mortas, mecanismos que detectam quando um conteúdo recebe seu último acesso, aplicando sobre ele alguma política de otimização, de forma a tornar a utilização das caches mais eficiente. No entanto, visto que as predições realizadas por estes mecanismos não são ideais, a adoção destas políticas pode prejudicar o funcionamento das caches, gerando consequências negativas para o sistema.

Este trabalho buscou estudar os efeitos causados por diferentes técnicas de otimização sobre dois preditores de linhas mortas de cache implementados sobre a *Last Level Cache* (LLC), os mecanismos *Dead Line and Early Write-Back Predictor* (DEWP) e *Skewed Dead Block Predictor* (SDP). Por meio da comparação entre duas técnicas de redução energética com efeitos positivos e negativos complementares sobre as caches, o *Gated-Vdd* e o *Drowsy*, pôde-se chegar à conclusão de que a técnica *Drowsy* é mais adequada quando aplicada isoladamente, ou seja, sem nenhuma técnica de otimização de desempenho complementar. Já o *Gated-Vdd*, por ser a técnica de otimização energética mais utilizada nos preditores de linhas mortas encontrados durante a presquisa bibliográfica, foi escolhido para avaliação junto às políticas de Prioridade e *Bypass*, revelando que, dependendo do objetivo esperado (energia ou desempenho), uma combinação distinta de técnicas é preferível ser aplicada. Além disso, nem sempre uma combinação efetiva para determinado preditor funcionará adequadamente para outro, devendo ser realizados estudos para descobrir as melhores combinações em cada mecanismo.

Os resultados deste trabalho sugerem que mecanismos e técnicas não são independentes, ou seja, nem sempre a combinação entre um preditor com boa acurácia e uma técnica de otimização eficiente resultarão em um bom desempenho. Assim, dependendo do mecanismo utilizado, uma combinação distinta de técnicas deve ser aplicada. Porém, se utilizadas em uma combinação adequada, essas técnicas possuem alto potencial para melhorar o desempenho dos sistemas, além de reduzir seus crescentes gastos com energia estática, fatores cruciais no desenvolvimento de novas gerações de processadores.

Alguns dos resultados obtidos durante a realização deste trabalho foram publicados no VIII Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC) (M. Sokulski et al., 2018) e no XIX Simpósio de Sistemas Computacionais de Alto Desempenho (WSCAD) (Sokulski et al., 2018). Esse artigo publicado no WSCAD foi selecionado como um dos *best papers* e convidado para extensão no periódico *Communications in Computer and Information Science* (CCIS), tendo sido recentemente aceito para publicação (M. Sokulski et al., 2019).

REFERÊNCIAS

Alves, M. A. Z., , K., Ebrahimi, E., Narasiman, V., Villavieja, C., Navaux, P. e N. Patt, Y. (2012). Energy savings via dead sub-block prediction.

Alves, M. A. Z. (2014). *Increasing Energy Efficiency of Processor Caches via Line Usage Predictors*. Tese de doutorado, Rio Grande do Sul Federal University, Porto Alegre, RS, Brazil.

Alves, M. A. Z., Villavieja, C., Diener, M. e Navaux, P. O. A. (2013). Energy efficient last level caches via last read/write prediction. Em *Int. Symp. on Computer Architecture and High Performance Computing*.

Block, H. D. (1988). Neurocomputing: Foundations of research. capítulo: The Perceptron: A Model for Brain Functioning. I, páginas 135–150. MIT Press, Cambridge, MA, USA.

Borkar, S. (1999). Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29.

Chang, K. K. (2017). *Understanding and Improving the Latency of DRAM-Based Memory Systems*. Tese de doutorado, Carnegie Mellon University, Pittsburgh, PA.

Faldu, P. e Grot, B. (2017). Leeway: Addressing variability in dead-block prediction for last-level caches. Em *Int. Conf. on Parallel Arch. and Compilation Techniques*.

Flautner, K., Kim, N. S., Martin, S., Blaauw, D. e Mudge, T. (2002). Drowsy caches: simple techniques for reducing leakage power. Em *Proceedings 29th Annual International Symposium on Computer Architecture*, páginas 148–157.

Gil, A. (2002). *Como elaborar projetos de pesquisa*. Atlas.

Intel (2018). Pin - a dynamic binary instrumentation tool.

Jain, A. e Lin, C. (2016). Back to the future: Leveraging belady's algorithm for improved cache replacement. Em *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, páginas 78–89.

Jeff Preshing, H. P. (2012). A look back at single-threaded cpu performance. `http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/`. Acessado em 15/07/2019.

Khan, S., Burger, D., Jiménez, D. A. e Falsafi, B. (2010). Using dead blocks as a virtual victim cache. Em *Int. Conf. on Parallel Arch. and Compilation Techniques*.

Kharbutli, M. e Solihin, Y. (2008). Counter-based cache replacement and bypassing algorithms. *IEEE Transactions on Computers*, 57(4).

Kim, N. S., Austin, T., Baauw, D., Mudge, T., Flautner, K., Hu, J. S., Irwin, M. J., Kandemir, M. e Narayanan, V. (2003). Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75.

M. Sokulski, R., D. Carreno, E. e Z. Alves, M. A. (2018). Evaluating dead line predictors efficiency with drowsy technique. Em *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, páginas 250–255.

M. Sokulski, R., D. Carreno, E. e Z. Alves, M. A. (2019). Evaluating cache line behavior predictors for energy efficient processors. *Communications in Computer and Information Science*. No prelo.

Muralimanohar, N., Balasubramonian, R. e Jouppi, N. P. (2008). Architecting efficient interconnects for large caches with cacti 6.0. *IEEE Micro*, 28.

Patil, H., Cohn, R. et al. (2004). Pinpointing representative portions of large intel ® itanium ® programs with dynamic instrumentation. Em *Int. Symp. on Microarchitecture*.

Powell, M., Yang, S.-H. et al. (2000). Gated-vdd: A circuit technique to reduce leakage in deep-submicron cache memories. Em *Int. Symp. on Low Power Electronics and Design*.

Raviraj, R. H. S. (2018). Spec cpu2017: Performance, energy and event characterization on modern processors. Dissertação de Mestrado, The School of Graduate Studies of The University of Alabama in Huntsville, Huntsville, Alabama.

Santos, P. C., Alves, M. A. Z., Diener, M., Carro, L. e Navaux, P. O. A. (2016). Exploring cache size and core count tradeoffs in systems with reduced memory access latency. Em *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, páginas 388–392.

Seznec, A. (1993). A case for two-way skewed-associative caches. Em *Int. Symp. on Computer Architecture*.

Sokulski, R., Carreño, E. e Alves, M. (2018). Introducing drowsy technique to cache line usage predictors. Em *2018 Symposium on High Performance Computing Systems (WSCAD)*, páginas 259–265.

Teran, E., Wang, Z. e Jimenez, D. A. (2016). Perceptron learning for reuse prediction. Em *Int. Symp. on Microarchitectur*.

Zhou, H., Toburen, M. C., Rotenberg, E. e Conte, T. M. (2003). Adaptive mode control: A static-power-efficient cache design. *ACM Trans. Embed. Comput. Syst.*, 2(3):347–372.