Arquitetura de Memória Cache Reconfigurável

Milene B. Carvalho[1,2], Carlos A. P. S. Martins[1]
[1]*Laboratório de Sistemas Digitais e Computacionais (LSDC)*
Pontifícia Universidade Católica de Minas Gerais, Av. Dom José Gaspar 500,
Belo Horizonte, MG.
[2]*Universidade de Itaúna, Faculdade de Engenharia, Rodovia MG 431 - Km 45, Itaúna, MG.*
[1]*milene@ieee.org,* [2]*capsm@pucminas.br*

Resumo

Neste artigo apresentamos uma arquitetura de memória cache reconfigurável. Além disso, apresentamos a taxa de acerto de algumas cargas de trabalho reais do SPEC 2000 CPF representadas por traces obtidos do BYU Trace Distribution Center executadas em nossa arquitetura e nas organizações tradicionais de cache. Com a análise dos resultados destes experimentos, foi possível concluir que com o mesmo número de comparadores, a cache reconfigurável possui uma taxa de acerto superior a uma cache fixa, na maioria dos casos. Nossa principal contribuição é a proposta de uma arquitetura de memória cache reconfigurável capaz de se adaptar às diferentes cargas de trabalho e a análise de seu comportamento.

1. Introdução

Diversas inovações arquiteturais têm sido desenvolvidas com o objetivo de melhorar o desempenho dos sistemas computacionais [1]. Um dos gargalos no processamento é o acesso à memória. Para reduzir o tempo gasto com este acesso, o conceito de memória cache vem sendo usado [2].

As características arquiteturais da cache determinam quais são as cargas de trabalho (em função de suas características) que terão melhor desempenho quando executadas [3]. No entanto, a arquitetura das caches presentes nos processadores atuais e seus parâmetros arquiteturais são baseados em uma configuração que possua um desempenho bom para uma carga de trabalho média, isto é, uma configuração que não é ideal para todas as cargas de trabalho do sistema, mas para uma média das mesmas. Como as caches são fixas, (quando a cache é projetada, sua configuração é definida e a arquitetura da cache não pode ser modificada após a sua fabricação) o desempenho das mesmas nem sempre é otimizado.

Através da adaptação da memória cache à carga de trabalho através de reconfiguração [4][5], seria possível melhorar o desempenho da mesma, adequando a estrutura e o funcionamento da cache à carga de trabalho específica executada a cada momento em um sistema computacional. Portanto, os objetivos desta pesquisa são propor, desenvolver, verificar e analisar a utilização de conceitos de reconfiguração no projeto de arquiteturas de memória cache e de adaptação da mesma à carga de trabalho (*workload*).

Este artigo apresenta os resultados finais da pesquisa [6] que gerou outros artigos [7][8], com resultados parciais, como os apresentados no WSCAD 2004. Desde a publicação deste, a pesquisa evoluiu e foi possível analisar não somente a eficácia da adaptação da memória cache à carga de trabalho, mas também as situações em que a adaptação aproxima o desempenho da cache reconfigurável ao limite máximo (cache completamente associativa) considerando a taxa de acerto [9]. Portanto, o objetivo secundário deste artigo é analisar o comportamento da cache reconfigurável e compará-lo com o comportamento de caches com organização fixa.

2. Trabalhos correlatos

Nesta seção apresentamos alguns trabalhos correlatos à nossa pesquisa. Os trabalhos relacionados ao problema, isto é, à melhoria do desempenho de memória cache são inúmeros [10]. Os trabalhos correlatos ao nosso problema e à nossa solução são os que usam de reconfiguração, adaptação ou variação da organização da memória cache como maneira de se obter uma melhora no desempenho da mesma.

Considerando a localidade espacial, existem trabalhos que apresentam mudanças no tamanho do bloco/linha e/ou no tamanho de busca (*fetch*) [11].

Considerando a localidade temporal, existem trabalhos que utilizam a cache para realizar outras tarefas, como a de auxílio no processamento [12][13] [14] e os trabalhos que modificam a associatividade da cache. Alguns trabalhos, como [15], além de usar técnicas como as citadas anteriormente, exploram a não necessidade do uso total da área do dispositivo destinada à cache para reduzir o consumo de energia.

Além disso, diversos esforços vêm sendo realizados para reduzir o consumo de energia através não só de características da carga de trabalho como também da tecnologia de implementação dos dispositivos de memória [16]. Nestes casos, a tecnologia de implementação das memórias pode ser usada em conjunto com nossa proposta em um trabalho futuro, já que nosso objetivo principal neste trabalho é o desempenho da memória cache.

Em [17], além da diminuição da associatividade da memória cache para reduzir o consumo de energia, ainda é possível "modificar" o número de *slots* da cache. Isto é feito através da concatenação dos *ways*. Portanto, reduzindo-se a associatividade desta cache, é possível aumentar o número de *slots* da mesma.

Uma organização de memória cache que se difere das tradicionais é a cache vítima (*victim cache*) [18]. Na realidade, esta cache é um pequeno *buffer* usado em conjunto com outra cache de organização tradicional. Este *buffer* guarda os blocos que recentemente foram retirados da cache principal. Desta maneira, a cache vítima funciona como um aumento da associatividade da mesma.

Em [11] um sistema de cache chamado DASAT (*Dynamically Aggressive Spatial and Adaptive Temporal*) é proposto. Este sistema é composto por duas caches no mesmo nível: uma do tipo mapeamento direto com um bloco pequeno e uma cache completamente associativa com bloco grande. O tamanho de *fetch* é variável dinamicamente, explorando a localidade espacial da carga de trabalho. Através de monitores, alguns blocos são mantidos na cache mapeamento direto através da análise da localidade temporal. Cada bloco grande da cache completamente associativa pode ser configurado como múltiplos blocos pequenos, garantindo flexibilidade ao sistema.

A *Reactive-Associative Cache* (r-a cache) [19] proporciona flexibilidade de associatividade. Ela possui dois tipos de posição para armazenamento de dados: mapeamento direto e associativo por conjunto. A segunda possui um tempo de acerto maior que a primeira. Para proporcionar flexibilidade e alto desempenho, nesta organização, os blocos são armazenados nas posições do tipo mapeamento direto e, somente em caso de conflito, os blocos são armazenados nas posições do tipo associativo por conjunto. Esta cache possui a vantagem das caches do tipo mapeamento direto em relação ao tempo de acerto se o dado desejado puder ser encontrado nessas posições. Além disso, ela não possui as desvantagens de conflito das caches mapeamento direto. No entanto, quando é necessário acessar dados conflitantes (armazenados nas posições do tipo associativo por conjunto), o tempo de acesso é maior que o normal (mapeamento direto).

3. Arquitetura de cache reconfigurável

Uma memória cache reconfigurável é uma cache que permite que sua estrutura seja modificada, diversas vezes, depois que a cache é fabricada.

Nossa arquitetura de cache reconfigurável é composta por dois módulos principais: a organização reconfigurável e a política de adaptação, Figura 1. A organização reconfigurável é composta pelo sub módulo de armazenamento de blocos da memória principal e pela lógica necessária para que estes blocos sejam acessados. A política de adaptação é o módulo responsável por determinar a configuração mais adequada dada uma carga de trabalho e por realizar a reconfiguração da estrutura do módulo organização reconfigurável, modificando o comportamento da memória cache.

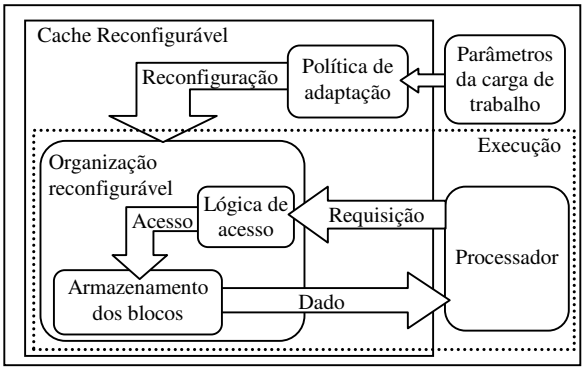

Figura 1. Arquitetura de cache reconfigurável

A política de adaptação tem como parâmetros de entrada, características da carga de trabalho do sistema e métricas de desempenho ou configurações da memória cache. Ela escolhe a configuração da cache que determinará seu comportamento, como se estivesse escolhendo uma entre as diversas configurações possíveis considerando as limitações da arquitetura

reconfigurável. Os parâmetros de entrada da política de adaptação podem ser indicados pelo usuário, estimados ou medidos com a execução da carga. Quanto mais confiáveis forem os parâmetros, melhor será a configuração definida pela política.

Além dos parâmetros de entrada, a própria lógica da política de adaptação influencia na qualidade das configurações determinadas pela política. Uma lógica simples pode não ter um bom resultado, no entanto, uma lógica complexa pode inviabilizar a implementação da política ou a sua utilização durante o funcionamento da cache.

Como os parâmetros de entrada e a lógica da política de adaptação influenciam na qualidade de sua solução, uma das maneiras de se obter bons valores para os parâmetros sem sobrecarregar a política é realizar o monitoramento da execução de uma carga de trabalho ou parte dela. Este monitoramento é realizado durante um dado instante de tempo, chamado *quantum*. Este deve ser pequeno o suficiente para não gerar muitas estatísticas, mas grande o suficiente para caracterizar o comportamento da carga. Quando o monitoramento é finalizado, a política de adaptação pode ser executada e uma nova configuração é escolhida.

Se considerarmos que este monitoramento é realizado diversas vezes e, conseqüentemente, a reconfiguração da organização da cache também, então teremos uma cache reconfigurável dinâmica. No entanto, se isso não for possível, os parâmetros podem ser determinados uma única vez para uma determinada carga e esta terá apenas uma configuração de cache a ela associada. Neste caso, consideramos que temos uma cache reconfigurável estática. Neste tipo de cache reconfigurável, a configuração é escolhida uma única vez, antes da execução da carga de trabalho.

Em sistemas computacionais multitarefa, diversos processos usam os recursos do sistema em espaços de tempo diferentes. Como geralmente o escalonador do sistema operacional determina um tempo que o processo pode usar os recursos e depois ele é colocado em uma fila para voltar a usar, não é uma boa prática usar uma única configuração de cache para todos estes processos. Portanto, cada configuração da cache deve ser associada a um processo, garantindo assim que a configuração seja adequada ao mesmo.

4. Cache com associatividade variável independente de *slot*

Nesta seção apresentamos a cache reconfigurável com associatividade variável independente de *slot* apresentada em [7]. Esta é uma das possibilidades de implementação da arquitetura de cache reconfigurável.

Diversos trabalhos, apresentados na seção 2, propõem que, de acordo com a localidade espacial da carga de trabalho, o tamanho do bloco lido da memória principal e gravado na cache seja variável para acompanhar as diferenças de localidade. Como já existem estes trabalhos que realizam a otimização da taxa de falta baseada na localidade espacial, neste trabalho, resolvemos descrever possíveis otimizações baseadas também na localidade temporal, para que as técnicas pudessem também ser usadas em conjunto, melhorando o desempenho das aplicações ainda mais.

Dentre os parâmetros que mantivemos constantes, estão o número de blocos possíveis de se armazenar na cache e o número de *slots*. Esta escolha é baseada no fato de que mudanças nesses dois parâmetros alterariam todo o projeto da memória cache, surgindo outros problemas. Como pretendemos realizar modificações na memória depois de sua fabricação ou até mesmo de maneira dinâmica, isto é, enquanto uma carga de trabalho é executada, tais parâmetros foram mantidos constantes.

Já se sabe que um aumento na associatividade de uma cache, geralmente, melhora o desempenho da mesma, porque evita faltas por conflitos [10]. Isso acontece porque se aumentando a associatividade, um bloco que poderia ser substituído e depois seria acessado novamente (causando uma falta) poderia continuar na memória cache. No entanto, isso somente é totalmente válido se considerarmos um aumento no tamanho da cache (não redução do número de *slots*) para não criar outras situações conflitantes. Estas situações ocorrem quando *slots* carentes por entradas na cache com associatividade menor passam a compartilhar um mesmo *slot* na cache com associatividade maior. Nesta situação, a taxa de falta geralmente não diminui e pode até aumentar.

Portanto, nossa proposta é modificar a associatividade da cache para diminuir o número de faltas por conflito sem, no entanto, aumentar o tamanho da cache (ou reduzir o número de *slots*) sem gerar outras situações conflitantes.

Denominamos uma cache reconfigurável A_i-A_f*way* uma cache que possui uma arquitetura reconfigurável, com associatividade inicial igual a A_i e associatividade máxima igual a A_f. Esta última indica o número máximo de entradas de um *slot* que podem ser verificadas simultaneamente, isto é, o número de comparadores da cache. Esta cache reconfigurável funciona como uma cache associativa por conjunto, mas não há a restrição de ter todos os *slots* com a mesma associatividade.

Uma cache reconfigurável *2-4way* é representada na Figura 2. Nesta figura, os blocos representados pela cor mais clara indicam os blocos fixos, isto é, os que não podem ser alocados para outros *slots* para que a cache continue com o mesmo número de *slots*. Os blocos mais escuros são os que são realocados pela política de adaptação. Neste caso, o primeiro *slot* recebeu dois novos blocos, enquanto os segundo e quarto *slots* doaram um bloco cada. O terceiro *slot* não sofreu alterações em sua associatividade.

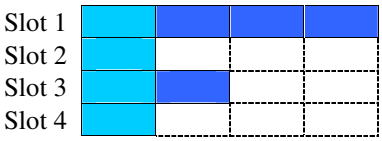

Figura 2. Cache reconfigurável 2-4way

5. Política de adaptação

Para realizar os experimentos com cargas de trabalho reais, projetamos uma política de adaptação simples, baseada no senso comum de que um *slot* com um número de faltas grande necessita de uma maior associatividade para reduzir este número. A qualidade dos resultados está diretamente relacionada com a implementação da política de adaptação.

Em nossa política implementada, consideramos dois parâmetros de entrada obtidos com a execução da carga de trabalho em uma organização de cache: número de acessos por *slot* e número de faltas por *slot*. O número de acessos indica o nível de utilização do *slot*, isto é, indica se uma mudança na associatividade dele altera o desempenho da cache ou se o desempenho não é muito influenciado por ele. O número de faltas por *slots* indica se a associatividade do mesmo é suficiente para suportar a demanda.

Nesta implementação, uma tabela estatística é preenchida a cada *quantum* da política. Esta tabela contém as métricas de desempenho usadas como parâmetros da política de adaptação. Além destas, outras duas métricas são obtidas ao final do *quantum* através das duas anteriores: a média de acessos por *slot* e a média de faltas por *slot*. De acordo com estas métricas, os *slots* são classificados como na Tabela 1.

Depois que os *slots* são classificados, são criadas duas listas, uma de *slots* doadores e outra de receptores. Na lista de receptores os *slots* GG ficam no início e são seguidos pelos *slots* GP. A ordem dos *slots* de um mesmo grupo é determinada pela ordem em que os mesmos são classificados. Então as listas são cruzadas e o primeiro receptor recebe um bloco do primeiro doador. Depois, ambos são removidos de suas listas. Este processo é repetido até que uma das listas fique vazia. Desta maneira, cada vez que um *quantum* termina, os receptores que forem atendidos terão sua associatividade aumentada em um e os doadores usados terão a associatividade reduzida em um.

Tabela 1. Classificação dos *slots*

Classe	Nº de faltas	Nº de acessos	descrição
GG	Grande	Grande	Possui maior prioridade para receber um bloco.
GP	Grande	Pequeno	Possui menor prioridade para receber um bloco.
PG	Pequeno	Grande	Condição ideal: não recebe nem doa blocos.
PP	Pequeno	Pequeno	Deve doar blocos para *slots* GG ou GP.

6. Experimentos

Para verificar e avaliar nossa proposta, tentamos aproximar ao máximo as especificações das caches e *traces* às usadas comercialmente. Por isso, utilizamos uma configuração de uma máquina comercial que possuímos. Simulamos uma cache de 64 KB para instruções e dados, com blocos de 4 palavras. A política de substituição de blocos da memória cache usada é a LRU (*Less Recently Used*).

As organizações de cache simuladas podem armazenar o mesmo número de blocos, 4096, com exceção da cache Double4way, que armazena o dobro de blocos. Portanto, nestes experimentos, um aumento na associatividade implica em uma diminuição no número de *slots*. As organizações simuladas e a relação entre elas, a associatividade e o número de *slots* estão representados na Tabela 2. Nesta tabela, N é o menor número de *slots* usados em nossas simulações (1024).

Usamos *traces* reais obtidos do *BYU Trace Distribution Center* [20]. Estes *traces* correspondem a execução de alguns *benchmarks* SPEC (*Standard Performance Evaluation Corporation*) executados utilizando-se o Windows 2000. Os *traces* disponíveis no *BYU Trace Distribution Center* possuem, além dos acessos à memória gerados pelo próprio *benchmark*, os acessos também do sistema operacional (SO). Nossa opção por usar este tipo de *trace* está baseada no fato de que nenhuma carga de trabalho, no contexto de nossa pesquisa, é executada sem a interferência de um sistema operacional, portanto, a existência dos acessos do SO faria com que o experimento ficasse mais real.

Tabela 2. Configurações de caches simuladas

Configurações de cache	Tamanho	Comparadores
2way	2 x 2N	2
2-4way	2 x 2N	4
2-8way	2 x 2N	8
4way	4 x N	4
Completamente associativa	4N x 1	4096
Double4way	4 x 2N	4

Para os experimentos, usamos traces do SPEC CFP (*SPEC CPU Float Point*) pertencentes ao SPEC CPU2000. Resultados de experimentos com o SPEC CINT (*SPEC CPU Integer*) foram apresentados em [7]. Os *traces* do CFP usados em nossos experimentos e o número de acessos à memória coletados para a simulação são apresentados na Tabela 3. O número de acessos é variável porque no *BYU Trace Distribution Center* são disponibilizados *traces* com cem milhões de referências, no entanto, para nossa simulação, filtramos o *trace* para que somente os acessos à memória de instruções e dados fossem simulados. Como este número de acesso é variável de carga para carga, o número de acessos simulados também é variável.

Tabela 3. Traces usados nos experimentos

Nome	Nº Acessos
applu	10.352.349
art	10.367.655
galgel	10.325.242
mesa	10.366.592
mgrid	10.363.192
swim	10.377.743
wupwise	10.319.347

Como simulamos *traces* de um sistema operacional multitarefa (Windows 2000), tivemos que considerar o escalonamento de processos realizado pelo SO. Por isso, a cada 10 ms do tempo de simulação, interrompíamos a mesma e invalidávamos o conteúdo da cache. Para reduzir o possível *overhead* imposto pela reconfiguração da cache, consideramos que o *quantum* da política de adaptação é igual a 10 ms.

7. Resultados

Nos gráficos das Figuras 3 a 7 são apresentadas as taxas de acerto dos *traces* usados em nossos experimentos. Ao analisar-se os gráficos é importante lembrar que quanto maior é a taxa de acerto, melhor será o desempenho da memória cache, se compararmos caches com o mesmo tempo de acesso e penalidade por falta.

Analisando-se os gráficos, pode-se afirmar que, a cache associativa por conjunto 4way possui uma taxa de acerto superior à da cache 2way. Isto mostra que com o aumento da associatividade os conflitos foram reduzidos não existindo conflitos entre *slots* da 2way que passaram a compartilhar *slots* na 4way a ponto de degradar o desempenho.

No entanto, analisando-se o gráfico da Figura 3, a distância entre a taxa de acerto da cache completamente associativa e as associativas por conjunto, mostra que ainda existem muitas faltas por conflito nestas organizações. Porém, a cache reconfigurável mostra-se como uma solução para reduzir o número de faltas por conflito. Por isso, como pode-se verificar no gráfico da Figura 3, que a taxa de acerto das caches reconfiguráveis 2-4way e 2-8way apresentam um desempenho melhor que as caches associativas por conjunto.

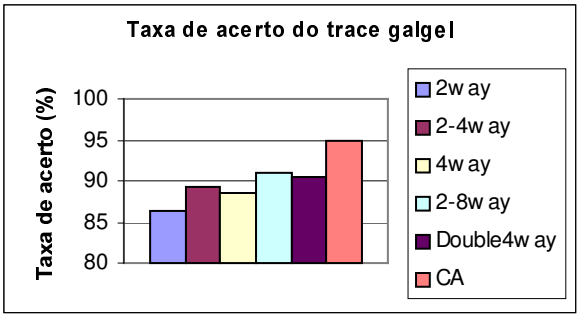

Figura 3. Taxa de acerto do trace galgel

A cache 2-4way consegue se adaptar à carga de trabalho obtendo melhor desempenho que a 4way. Esta melhora é obtida porque a cache reconfigurável 2-4way usa o número de *slots* da cache 2way (evitando o compartilhamento de *slots* por *slots* conflitantes, como ocorre na 4way) e oferecendo blocos aos *slots* da 2way carentes em blocos (chegando à associatividade 4). A cache reconfigurável 2-8way possui as mesmas características que a 2-4way e ainda a possibilidade de *slots* com associatividade até 8. Como o desempenho da 2-8way foi melhor que o da 2-4way, podemos afirmar que existem *slots* com necessidade maior que 4 blocos. Como a cache reconfigurável 2-8way oferece a possibilidade de mais de 4 comparadores em um *slot*, esta cache teve o melhor desempenho entre as caches com associatividade restrita.

A cache 2-8way possui melhor desempenho até mesmo em relação à cache associativa por conjunto Double4way. O baixo desempenho da cache Double4way mostra que, mesmo aumentando-se a capacidade da memória cache, não foi possível aumentar significativamente a taxa de acerto,

mostrando que existe um número significativo de faltas por conflito. Como a cache 2-8way tem a capacidade de reduzir o número de faltas por conflito, ela conseguiu um desempenho melhor que o da Double4way, mesmo possuindo a metade da capacidade da Double4way. No entanto, a diferença entre a taxa de acerto da cache completamente associativa em relação às demais, mostra que ainda existem muitas faltas por conflito. Uma otimização na política de adaptação da memória cache reconfigurável provavelmente melhoraria o desempenho das caches reconfiguráveis.

Analisando-se o gráfico da Figura 4, pode-se afirmar que a distância entre a taxa de acerto da cache completamente associativa e as associativas por conjunto, mostra que ainda existem muitas faltas por conflito nestas organizações.

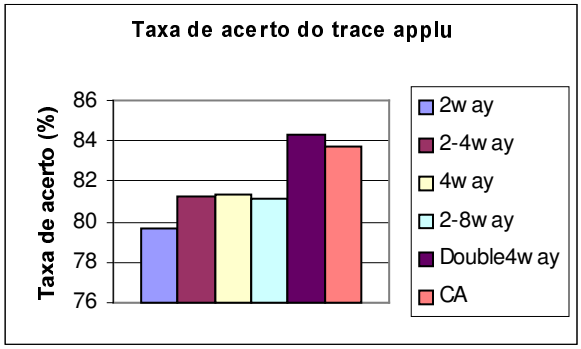

Figura 4. Taxa de acerto do trace applu

Diferentemente da execução do trace galgel, a taxa de acerto das caches reconfigurável 2-4way e 2-8way são muito semelhantes e inferiores à da cache 4way. Isto pode ter acontecido por dois motivos ou pela união destes dois. O primeiro motivo é a não necessidade de uma associatividade máxima superior a quatro na cache reconfigurável, usando esta política. O outro motivo pode estar associado à política de adaptação. Esta política demora mais para se adaptar à uma mudança na carga de trabalho quando a associatividade é maior. Com isso, os possíveis ganhos de se ter uma associatividade maior podem ser mascarados por uma perda gerada pelo tempo necessário para que a política de adaptação consiga adequar a cache reconfigurável 2-8way à carga de trabalho. A demora da política de adaptação para adequar a cache à carga de trabalho pode explicar a proximidade dos valores da taxa de acerto das caches reconfiguráveis e da associativa por conjunto 4way.

Comparando-se a taxa de acerto das caches completamente associativa e Double4Way e a taxa de acerto das demais caches, pode-se afirmar que uma melhora de desempenho destas últimas pode ser atingida através da redução das faltas por conflito e/ou por capacidade. Com a redução das faltas por capacidade, é possível se aproximar do desempenho da cache Double4way ou até superá-lo. Através da redução das faltas por conflito, é possível aproximar do desempenho da cache completamente associativa. Reduzindo-se os dois tipos de falta, é possível obter uma taxa de acerto superior a todas as taxas apresentadas na Figura 4.

Analisando-se o gráfico da Figura 5, pode-se afirmar que ainda existem muitas faltas por conflito nas caches associativas por conjunto. A taxa de acerto da cache reconfigurável 2-4way é inferior à da cache associativa por conjunto 4way, mostrando que os problemas da política de adaptação já citados anteriormente não permitiram que fosse possível melhorar o desempenho através das vantagens oferecidas pela associatividade variável independente de *slot*.

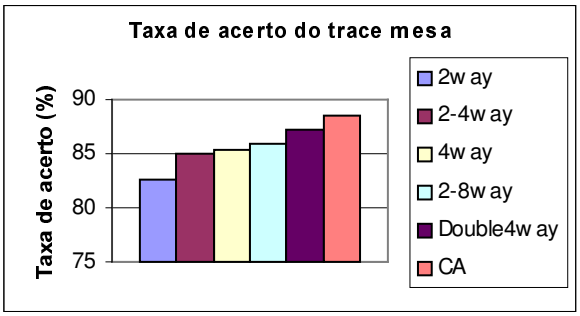

Figura 5. Taxa de acerto do trace mesa

A cache reconfigurável 2-8way, apesar de sofrer com os mesmos problemas da política de adaptação que a cache 2-4way sofre, apresentou um desempenho superior ao dela e ao da 4way. Isso mostra que o trace mesa possui um desempenho melhor quando executado em uma cache com alguns *slots* com associatividade máxima igual a oito. A diferença entre as taxas de acerto da cache reconfigurável 2-8way e da completamente associativa indica que ainda existem diversas faltas por conflito, no entanto, algumas adequações na política de adaptação poderiam diminuir esta diferença significativamente. A diferença entre a taxa de acerto das caches associativa por conjunto Double4way e completamente associativa mostra que existem muitas faltas por conflito na Double4way, porque, mesmo tendo o dobro do tamanho das demais caches, seu desempenho foi pior que o da cache completamente associativa. Isso mostra que o número de faltas por conflito foi maior que o número de acertos que antes eram faltas por capacidade.

Analisando-se o gráfico da Figura 6, pode-se afirmar que o desempenho das caches que possuem limite de associatividade poderia ser melhorado tanto pelo aumento da capacidade quanto pela redução das faltas por conflito.

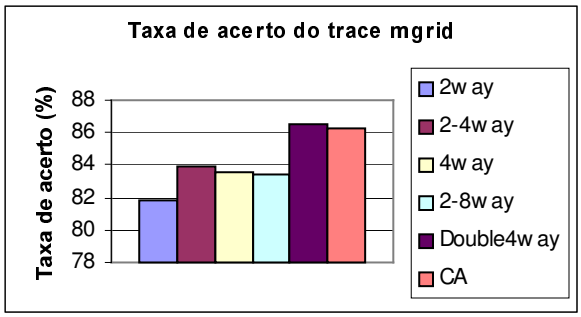

Figura 6. Taxa de acerto do trace mgrid

Analisando-se a taxa de acerto das caches reconfiguráveis 2-4way e 2-8way e a da cache associativa por conjunto 4way, pode-se dizer que, através da utilização da reconfigurabilidade de associatividade, é possível obter um desempenho melhor que uma cache com associatividade fixa considerando o mesmo número de comparadores (cache reconfigurável 2-4way e cache associativa por conjunto 4way). O desempenho da cache 2-8way foi pior que o da 4way, possivelmente, por causa dos problemas da política de adaptação explicados anteriormente: tempo gasto até o final do *quantum* (quando ocorre a reconfiguração e conseqüente adaptação à carga) e dificuldade de adequação a mudanças ocorridas na carga quando a associatividade de alguns *slots* é alta. A redução do impacto do primeiro problema da política de adaptação citado sobre o desempenho da cache pode melhorar o desempenho da cache 2-4way também.

Analisando-se o gráfico da Figura 7, pode-se dizer que os *traces* art, swim e wupwise possuem taxas de acerto com comportamento parecido com o do trace galgel, apresentado na Figura 3. Portanto, as análises feitas para o trace galgel são válidas para estes traces, com exceção dos comentários feitos em relação à cache Double4way. Analisando a taxa de acerto obtida com a execução dos *traces* na cache Double4way, pode-se verificar, principalmente para o *trace* wupwise que é possível melhorar o desempenho da memória cache não só diminuindo o número de faltas por conflito, mas também aumentado a capacidade (diminuindo as faltas por capacidade).

Através da adaptação da cache à carga de trabalho, foi possível, com uma memória cache reconfigurável com os mesmos números de comparadores e entradas de *slot* de uma cache com organização fixa, obter uma taxa de acerto maior que a da cache fixa. Isso mostra que, sem aumentar o número de comparadores e/ou o tamanho da memória cache, é possível diminuir o número de faltas por conflito, utilizando a cache reconfigurável. O aumento no número de comparadores utilizados (2-8way) também causou um aumento na taxa de acerto na maioria dos experimentos.

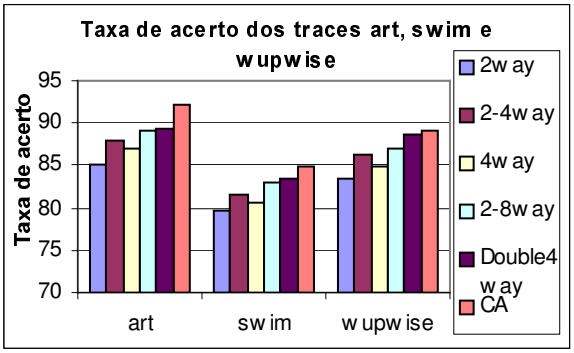

Figura 7. Taxa de acerto dos traces art, swim e wupwise

8. Conclusão

Neste artigo, propusemos uma arquitetura de memória cache reconfigurável verificada com uma organização com associatividade variável independente de *slot*. Apesar das limitações da política de adaptação conseguimos bons resultados.

Concluímos que a flexibilidade e a possibilidade de se ter *slots* com associatividade diferenciada e variável em uma mesma memória cache permite que a organização da memória seja adaptada às características de cada carga de trabalho.

Através da adaptação da cache à carga de trabalho, foi possível, com uma cache reconfigurável com os mesmos números de comparadores e entradas de *slot* de uma cache com organização fixa, obter uma taxa de acerto maior que a da cache fixa, na maioria dos traces simulados. Isso mostra que, sem aumentar o número de comparadores e/ou o tamanho da memória cache, é possível diminuir o número de faltas por conflito.

Comparando-se os resultados obtidos nesta pesquisa com os apresentados em [7], foi possível verificar que a melhora de desempenho com a utilização da cache reconfigurável para os *traces* do SPEC CFP é maior que a do SPEC CINT. Esta diferença é explicada pelo maior número de acessos do CFP (possibilitando a política de adaptação um tempo maior para adaptar a cache) e pelas características das cargas de trabalho.

Como trabalhos futuros podemos citar a implementação de uma política de adaptação que considera outras características da carga de trabalho e o desenvolvimento de uma cache reconfigurável com outros parâmetros variáveis além da associatividade e a análise do impacto do tamanho do *quantum*.

Referências

[1] Patterson, D. A., Hennessy, J. L. Organização e Projeto de Computadores – A Interface Hardware/Software, 3a Edição, Editora Campus, 2005.

[2] Smith, A. J. Cache Memories, ACM Computing Surveys, 14(3):473-530, 1982.

[3] Jain, R. K. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling, John Wiley & Sons, 1991.

[4] Compton, K., Hauck, S. Reconfigurable Computing: A Survey of Systems and Software, ACM Computing Survey, 34(2):171-210, 2002.

[5] Martins, C. A. P. S., Ordonez, E. D., Corrêa, J. B. T., Carvalho, M. B. Computação Reconfigurável: conceitos, tendências e aplicações. In: XXII Jornada de Atualização em Informátcia (JAI), SBC2003, Volume 2, pp.339-388, 2003.

[6] Carvalho, M. B. Proposta e Desenvolvimento de uma Arquitetura de Memória Cache Reconfigurável. Dissertação apresentada ao Programa de Pós Graduação em Engenharia Elétrica da PUC Minas, 2005.

[7] Carvalho, M. B., Martins, C. A. P. S., Arquitetura de Cache com Associatividade Reconfigurável, Workshop em Sistemas Computacionais de Alto Desempenho, 2004.

[8] Carvalho, M. B., Góes, L. F. W. Martins, C. A. P. S., Dynamically Reconfigurable Cache Architecture Using Adaptive Block Allocation Policy. 13th Reconfigurable Architectures Workshop (RAW 2006), IPDPS 2006, 2006.

[9] Lennerstad, H., Lundberg, L. Optimal Worst Case Formulas Comparing Cache Memory Associativity. SIAM Journal on Computing, Volume 30, Issue 3, pp. 872–905, 2000.

[10] Hennessy, J. L., Patterson, D. A. Arquitetura de Computadores: Uma Abordagem Quantitativa, 3 a Edição, Editora Campus, 2003.

[11] Lee, J., Kim, S., Weems, C. Application-Adaptive Intelligent Cache Memory System, ACM Transactions on Embedded Computing Systems, pp. 56–78, 2002.

[12] Ranganathan, P., Adve, S., Jouppi, N.P. Reconfigurable Caches and Their Application to Media Processing, Proceedings of the International Symposium on Computer Architecture, pp. 214–224, 2000.

[13] Kim, H., Somani, A.K.,Tyagi, A. Adaptive Balanced Computing (ABC) Microprocessor using Reconfigurable Functional Caches (RFCs), Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02), 2002.

[14] Tanaka, K., Fukawa, T. Highly Functional Memory Architecture for Large-Scale Data Applications, Proceedings of the IEEE Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04), 2004.

[15] Sangireddy, R., Kim, H., Somani, A.K. Low-power high-performance reconfigurable computing cache architectures, IEEE Transactions on Computers, Volume 53, Issue 10, pp. 1274-1290, 2004.

[16] Kim, C. H., Kim, J., Mukhopadhyay, S., Roy, K. A Forward Body-Biased Low-Leakage SRAM Cache: Device, Circuit and Architecture Considerations, IEEE Transactions on Very Large Scale Integration Systems, 2005.

[17] Zhang, C., Vahid, F., Lysecky, R. A Self-Tuning Cache Architecture for Embedded Systems, Special Issue on Dynamically Adaptable Embedded System, ACM Transactions on Embedded Computing Systems, 2004.

[18] Jouppi, N. P. Improving direct-mapped cache performance by the addition of a small fully associative cache and prefetch buffers, Proceedings of the International Symposium on Computer Architecture, pp. 364–373, 1990.

[19] Batson, B., Vijaykumar, T. N. Reactive-associative caches, Proceedings of IEEE International Conference on Parallel Architectures and Compilation Techniques, pp. 49-60, 2001.

[20] BYU Trace Distribution Center. Disponível em: http://tds.cs.byu.edu/tds/, 2001.