Simulações de hierarquia de memória com o Simulador Amnésia

1st Davi Montalvão Ferreira
Institutos de ciências exatas e informáticas (ICEI)
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Brasil
davi.ferreira.1137912@sga.pucminas.br

2nd Henrique Temponi Maia
Institutos de ciências exatas e informáticas (ICEI)
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Brasil
henrique.maia.1135188@sga.pucminas.br

***Resumo*—Com esse trabalho esperamos aprofundar os nossos conhecimentos em hierarquia de memória, mais especificamente, memória principal e cache. E suas propriedades como: associatividade, tamanho de bloco e memoria e politicas de escrita. Esse artigo visa mostrar, pelo o simulador amnesia, e analisar os possíveis diferentes tipos de arquiteturas.**

***Index Terms*—Amnésia, hierarquia de memória, simulação, cache**

I. INTRODUÇÃO

Conceitos mais avançados requerem mais abstração, e na computação não é diferente a utilização de simuladores pode ajudar em esclarecer duvidas e melhorar o conhecimento. Um simulador possibilita exemplificar problemas ainda mantendo o foco no aprendizado no aluno. segundo Moreno [1] 90% dos estudantes que usaram um simulador de hierarquia de memoria, falaram que ajudou tanto no esclarecimento da matéria, e 70% deles o simulador ajudou a manter o seu interesse na matéria. Com isso em mente usamos um simulador chamado amnesia para testar diferente tipos de arquiteturas no intuito de aplicar e fortalecer nossos conhecimentos teóricos de hierarquia de memoria. Foram abordados diferente conceitos teóricos, como: espacialidade temporal e espacial, tamanho da cache, politicas de substituição, politicas de escrita e níveis de associatividade.

Com isso esse artigo tem em mente simular arquiteturas no amnesia gerando resultados em forma de gráficos para cada arquitetura. As outras partes desse artigo aborda as propostas de arquitetura, no qual inclui tabela mostrando as arquiteturas usadas e uma breve descrição do que foi simulado. Depois uma avaliação dos resultados onde os dados serão revisados e comparados com as expectativas vistas em sala.

II. TRABALHOS CORRELATOS

Para essa sessão de trabalhos correlatos procuramos artigos de conteúdo similhares ao apresentado.

O primeiro artigo de Moreno [1] faz uma pesquisa com um simulador chamado MNEME que trata hierarquia de memoria, o simulador foi introduzido a cursos de engenharia de computação e a ciencia de computação e permaneceu em fase de teste por 2 anos, nesses mesmos cursos, coletando *feedback* para melhorara o simulador, o simulador previne diferentes ferramentes para a fixação de conceitos complexos. Ao final do experimento foi constato que existe uma melhora no desempenho dos alunos ao desrespeito em melhorar aprendizado do conteúdo visto e ate uma melhora no entusiamo ao mesmo.

O segundo artigo de Lioris [2] fala sobre XMSIM um determinador de hierarquia de memoria para aplicações multimídia. Com isso é apresentado um código fonte, geralmente em c, e depois o simulador executa o programa e mostrando os diferentes níveis de cache. O artigo entra em detalhe como é a estrutura do XMSIM e como é organizado os dados, possíveis comandos, logo mostrando algumas telas do processo acontecendo. Por final ele comenta que com isso é possível melhorar o nível de otimização.

O terceiro artigo de Mei [3] cita como as fabricantes de GPU's (graphics processing units) não publicam detalhes da hierarquia de memoria de seus produtos, sendo a mesma um dos fatores cruciais para utilizar completamente o poder computacional da placa gráfica. Sendo assim os pesquisadores propuseram uma abordagem de microbenchmark de 3 gerações de placas da nvidia, especificamente investigando os diferentes sistemas de caches e latências de acesso. A objetivo final é ajudar a otimizar os softwares para as arquiteturas testadas.

O quarto artigo de Shoushtari [4] propõe uma arquitetura de sistema para software-assisted memory (SAM) hierarchy em sistemas de multiplos nucleos, foi visado superar a inflexibilidade e preço de pura hierarquia de memoria gerenciada por hardware. Os trabalhos preliminares mostraram oportunidades para salvar custos energeticos e aprimorar a performance utilizando: Uma Software Programmable Memory SPM/adaptável memoria de cache para aplicações, gerenciamento de data também via SPM e virtualizando e compartilhando espaço de memoria entre aplicativos. Os experimentos reduziram o tempo de execução e consumo de energia em 23% e 7% respectivamente, dependendo do tipo de trabalho.

Os artigos supracitados contem melhorias em otimização de hierarquia de memoria, exceto o primeiro, que possui maior foco no ensino do tema para diversos alunos via um simulador. Por outro lado o nosso trabalho foca numa simulação de arquiteturas no simulador amnesia, aonde visamos aplicar técnicas de hierarquia de memoria feita pelos participantes do grupo para desenvolver um modelo que apresentasse desempenho superior no nosso trace.

III. Proposta de Arquitetura

A proposta de arquitetura que o grupo definiu consiste em utilizar uma memoria principal de tamanho 32.

Tabela I
ARQUITETURAS DE TAMANHO DE CACHE 4

	modelo 1	modelo 2	modelo 3	modelo 4
bloco/linha	1	1	4	2
Cache	N/A	4	4	4
PDE	N/A	WT	WT	WT
PDS	N/A	FIFO	FIFO	FIFO
TDC	N/A	UNI	UNI	UNI
NDA	N/A	1	1	2

Tabela II
ARQUITETURAS DE TAMANHO DE CACHE 8

	modelo 5	modelo 6	modelo 7
bloco/linha	2	4	8
Cache	8	8	8
PDE	WB	WB	WB
PDS	FIFO	FIFO	FIFO
TDC	UNI	UNI	UNI
NDA	4	2	1

Tabela III
ARQUITETURAS DE TAMANHO DE CACHE 16

	modelo 8	modelo 9	modelo 10	modelo 11
bloco/linha	8	16	4	2
Cache	16	16	16	16
PDE	WB	WB	WB	WB
PDS	FIFO	FIFO	FIFO	FIFO
TDC	UNI	UNI	UNI	UNI
NDA	2	1	4	8

A tabelas acima mostra as combinações foi utilizada nos testes, sendo o modelo uma variação de 3 arquiteturas de controle, em relação ao tamanho da cache. Apartir desse controle foram implementadas mudanças para melhorar o hitrate da cache. Também na tabela apresenta bloco/linha, esse parâmetro é a quantidade do bloco/linha na cache e memoria principal, no simulador amnesia o tamanho do bloco precisa necessariamente ser igual o tamanho da linha. Depois temos o tamanho da cache, esse parâmetro indica o tamanho total para dados disponível na cache. O PDE (politica de escrita) indica qual politica de escrita tomara efeito no evento de miss, para fazer a sincronização entre a memoria principal e a cache. O PDS (politica de substituição) indica qual politica de substituição acontecera no evento da falta de memoria para colocar um novo dado na cache, escolhendo qual dado precisa ser substituído, foi usado WT (Write through) e WB (Write Back). O TDC (Tipo de cache) é um dos parâmetros disponível do simulador amnesia, estão disponíveis dois tipos de cache, o unified, onde o endereço e dados ficam em uma cache só, e o split, onde endereço e dados ficam em cache separadas, foi usado UNI (unified) e SPLT (splitted). e por ultimo NDA (nível de associatividade) que indica o nível de associatividade para dada arquitetura.

Outro aspecto importante é o trace, esse que é uma sequencia de instruções para ilustrar um possível programa ou uma rotina normal envolvendo a cache. O amnesia providencia instruções bem simples de apenas leitura, gravação e busca. Visto isso foi decidido pegar o código abaixo e escreve-lo, tentando aproxima-lo o mais possível com o código real.

```
double x = 0;
double y = 1.5/1000;
double z = 1 + y;
double c = 1;
int a = 10;
int b = 2;

for (int i = 0; i < b; i++) {
    y = y + a;
    y = y * (z ** c);

}
```

Como o código trata com variáveis, foi colocado que o double iria ocupar 4 espaços na memoria, e o integer apenas 2. As operações de aritméticas não foram consideradas que ocupavam espaço na memoria principal, apenas a operação de elevado (representado por **) foi colocada em memoria. Isso foi feito para estimular um pouco a questão de localidade temporal. O for que é apresentado no código é rodado apenas duas vezes, para o trace não ficar grande.

Visto isso o trace na parte inicial fica com varias sequencias, que esta lendo e gravando pela a parte do código, logo em seguida as instruções já ficam um pouco mais aleatórias pegando valores passados e usando tanto a localidade temporal e espacial.

O tempo por ciclo foi utilizado os valores padrões do amnesia sendo eles:

Tabela IV
TEMPO DE CICLOS

Mem/config	Ciclos por leitura	Ciclos por escrita	tempo por ciclo
cache	1	2	1
Principal	1	2	10

IV. Avaliação dos Resultados

todas as arquiteturas propostas possuem um tamanho de 32 na memória principal

Modelo 1: Sem cache

Usamos as arquiteturas presentes na amnésia como base, nesse caso sem a presença de um cache, para realizarmos uma referência para os modelos seguintes no trabalho proposto. Como demonstrado na tabela, esse modelo possui uma política de WriteThrough e não alteramos o método de substituição padrão (First In First Out - FIFO).

Modelo 2: block size 1

Figura 1. Caption

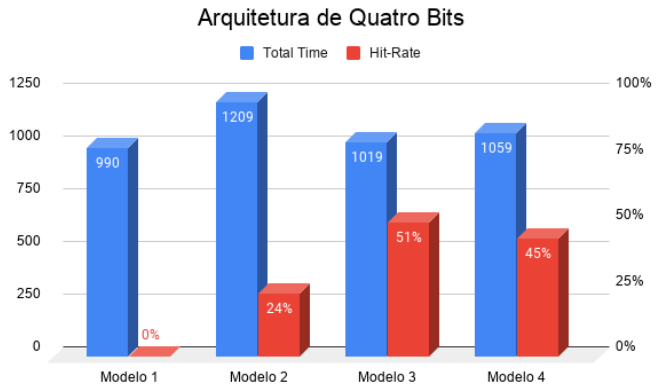
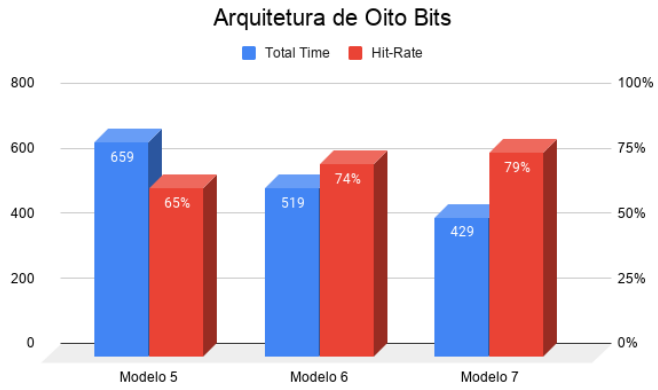


Figura 2. Caption



Apesar da velocidade do cache via mapeamento direto ser notoriamente superior à da memória principal, a alta taxa de miss resultou no modelo 2 sendo significativamente mais lento que o modelo 1. Isso ocorre devido a constante necessidade de acessar a memória principal e alocar a data no cache, um efeito causado pelo mau aproveitamento da localidade espacial. Daqui pra frente todas as arquiteturas possuirão cache do tipo unificado, incluindo essa.

Modelo 3: Block size 4

No modelo 3, também formado por mapeamento direto, o tamanho expandido no bloco aumentou o hit-rate da arquitetura, já que a probabilidade do dado estar presente no cache aumentou, aproveitando mais a localidade espacial nesse modelo. Apesar disso, o modelo 1 ainda se demonstrou superior no que diz respeito ao tempo gasto,uma indicação de que a arquitetura, ao menos para o programa testado, ainda está ineficiente.

Modelo 4: Block size 2 e Associative level 2

Nesse modelo tentamos uma abordagem associativa por conjunto, com o tamanho do bloco reduzido. Houve uma ligeira perda de desempenho em relação ao modelo 3 pois, ao diminuirmos o tamanho do bloco, o aproveitamento da localidade temporal. Visto que o tamanho da cache ainda não consegue se beneficiar totalmente da localidade espacial, para os modelos seguintes propomos uma expansão na mesma.

Modelo 5: Block Size 2 e Associative Level 4

Além do maior cache, optamos por uma política de escrita WriteBack em todos os modelos a seguir, incluindo esse. Aqui usufruímos de um nível associativo maior que aliado ao cache expandido, resultou pela primeira vez um modelo tendo um desempenho superior ao da referência (Modelo 1). O aumento do cache principalmente melhorou o hit-rate, ou seja, explorou adequadamente a localização temporal e aprimorou os resultados. Deve-se notar que boa parte dessa melhoria também é resultado da nova política de cache.

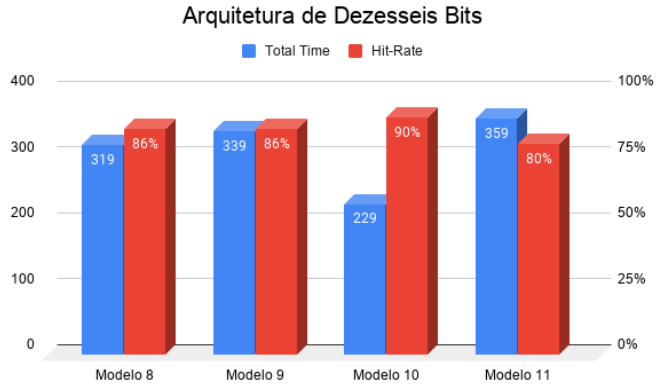
Modelo 6: Block Size 4 e Associative Level 2

Nossas observações implicaram que o programa iria se beneficiar mais de um bloco expandido do que um grande valor associativo, então foi colocado o size em 4 para explorar melhor a localidade espacial. O hit-rate expandiu outra vez e o desempenho, por consequência, sofreu melhorias.

Modelo 7: Block Size 8 e Associative Level 1

Por fim optamos por um mapeamento direto com um bloco do tamanho do cache, até então tivemos o melhor resultado, como observado nos gráficos dos modelos de 8 bits, que apresentam uma curva de melhoria estável. Até então presumimos que aproveitar a localidade espacial se demonstra mais benéfico para o desempenho do trace utilizado.

Figura 3. Caption



Modelo 8: Block Size 8 e Associative Level 2

Devido aos benefícios claros da expansão do cache outra vez optamos por dobrá-lo. Nesse modelo tentamos observar se o mesmo tamanho de bloco do modelo 7 com uma associação conjunta iria beneficiar ou prejudicar o desempenho, julgando assim se o programa funciona de forma superior em mapea-

mento direto na maioria dos casos. Inferimos então, baseado nos dados produzidos, que esse não é o caso, pois outra vez tivemos uma melhora na performance do programa.

Modelo 9: Block Size 16 e Associative Level 1

Contudo, achamos válido observar se outro bloco de mesmo tamanho do cache iria trazer benefícios semelhantes que o modelo 7 trouxe em relação ao 6, focando num maior aproveitamento da localidade espacial, via mapeamento direto. Sendo o modelo 9, o primeiro desde o modelo 2, a apresentar queda de desempenho em relação a arquitetura antecessora, apesar de possuir hit-rate similar. Concluímos então que os benefícios dessa abordagem nesse programa eventualmente chega a um ponto de retorno negativo, refletindo os estudos na aula de relação entre tamanho e efetividade.

Modelo 10: Block Size 4 e Associative Level 4

Com os dados obtidos nos modelos supracitados, chegamos a conclusão que a melhor abordagem para esse trace seria um balanceamento entre a exploração da localidade espacial e da temporal, via associação por conjunto. Os resultados refletiram a previsão, sendo o modelo 10 o de maior desempenho entre todos. Como esse programa possui muitas variáveis ocupando grande parte da memória, uma divisão via associação por conjunto com blocos que tendem a pegar apenas as variáveis mais utilizadas, trouxe melhorias na performance.

Modelo 11: Block Size 2 e Associative Level 8

Esse modelo foi criado apenas por prova de conceito, testando se as conclusões obtidas iriam bater com um teste real, o que ocorreu, pois um bloco menor não consegue explorar bem a localidade espacial e induz a uma maior porcentagem de miss, pelo menos no trace utilizado.

V. Conclusões

No simulador amnesia, o estudo de hierarquia de memoria é algo essencial para um bom aproveitamento, tanto do próprio computador,tanto quanto da cache. Por meio dele os tempos gerais de execução são reduzidos consideravelmente, possibilitando uma melhor comunicação entre o processador e o resto do computador. Além disso o simulador providenciou um melhor conhecimento sobre o assunto dado, dando um exemplo *hands-on* no assunto. Também como os dados são mais maleáveis temos um possível espaço para otimização, já que se pode gerar varias arquiteturas dependendo da situação a ser analisada. Usando os dados e os gráficos gerados podemos falar na importância e flexibilidade de arquiteturas utilizando conceitos de localidade espacial e temporal, pelo ajuste de tamanhos de blocos, e o nível de associatividade, para achar um possíveis combinações de arquitetura favoráveis para dado problema.

VI. Trabalhos futuros

Como esse trabalho deixou algumas possibilidades de arquiteturas para serem feitas, os próximos trabalhos poderão ser realizados utilizando as arquiteturas mudando mais parâmetros como: tipo de substituição, cache e talvez ate mesmo o tipo de escrita para determinar qual a porcentagem de melhoria para cada. Além disso podemos fazer testes com níveis variados de cache, e testes usando memoria virtual; temas esses que são de cunha importância para o conceito de hierarquia de memoria, possibilitando até mesmo um aumento do desempenho e do trace.

Referências

[1] Moreno, Lorenzo, et al. “MNEME: A memory hierarchy simulator for an engineering computer architecture course.” Computer Applications in Engineering Education, 19(2), 2011, pp.358-364.

[2] Lioris, Theodoros, Grigoris Dimitroulakos, and Konstantinos Masselos. “An early memory hierarchy evaluation simulator for multimedia applications.” Microprocessors and Microsystems 38.1, 2014, pp.31-41.

[3] X. Mei and X. Chu,“Dissecting GPU Memory Hierarchy Through Microbenchmarking,” in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 1, pp. 72-86, 1 Jan. 2017, doi: 10.1109/TPDS.2016.2549523.

[4] M. Shoushtari and N. Dutt, “SAM: Software-Assisted Memory Hierarchy for Scalable Manycore Embedded Systems,” in IEEE Embedded Systems Letters, vol. 9, no. 4, pp. 109-112, Dec. 2017, doi: 10.1109/LES.2017.2748098.