Hierarquia de Memória, uma análise dos algorítimos e localidades implementadas.

1st Eric Azevedo de Oliveira
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte, Brasil
quebec.ericjs@gmail.com

***Abstract*—O marco da computação, foi a criação da memória cache e todas as melhorias e formas de trabalhar com ela, sendo eles algorítimos como: *FIFO*(First In, First Out), *LRU*(Least Recently Used) , e medidas de localidades espacial e temporal, nas quais, se não foram utilizadas de forma correta, podem acarretar mais problemas que soluções. Partindo desse ideia, esse artigo irá decorrer de todas as formas de melhorias dadas a cache, apresentando qual a melhor situação para uso de cada algorítimo ou localidade, fazendo os testes aumentando o número de instruções e vendo qual o algorítimo irá sair melhor que o outro, e também aumentar as vias presentes na (cache) (Temporal),e o tamanho da linha (Espacial), visando avaliar como esses algoritmos atuariam em tais situações. Desta forma, terei uma visão mais clara do quanto a modificação do hardware bem como das instruções executadas terão um impacto significativo de cache(Miss, Page fault e Miss penalty).**

I. Introdução

A Cache é a penúltima camada na piramide de memória, com isso tem um acesso rápido, por estar localizada no próprio processador, tendo a intenção de reduzir o acesso do processador à memória principal, no qual demanda um tempo elevado para levar as informações até o processador, no qual gera uma referência da localidade contidua memória principal, existindo um limite de quantos dados podem ser armazenados. Desta forma é necessária uma política de armazenamento para reocupar espaço quando necessário.

Portanto, com a busca de otimizar os processos que são executados, no tempo mais curto possível, a priori aumentar o tamanho somente da cache, trabalhando com a localidade temporal, seria uma solução, entretanto não é viável, pelo alto custo de ter uma cache maior. Com isso atuar na memória principal também seria uma solução adequada, utilizando o TLB. Mas como o maior objetivo é achar a informação na cache, o principal foco será minimizar o custo e maximizar a otimização da cache.

Com isso em mente, por meio do simulador amnésia, pude averiguar qual desempenho, relacionado a localidade temporal e espacial, nos algorítimos de substituição do LRU e FiFO. Objetivando minimizar o cache miss, e por fim analisar o TBL, como medida de reduzir o tempo de acesso na memória RAM, foram feitos análises, dos algorítimos e métodos supracitados. Estando este artigo organizado da seguinte maneira:
Seção [2] irá mostrar pesquisas correlatas, apresentando de maneira breve o assunto de cada e como elas me ajudaram com o intendimento sobre a cache e memória principal.
Seção [3] apresenta as arquiteturas utilizadas para a realização dos testes, assim também com os cenários.
Resultados experimentais são colocados na Seção [4]. Por fim, a seção 5 dá conclusão da minha pesquisa.

II. Trabalhos Correlatos

O artigo [2], diz muito a respeito dos algorítimos LRU e FIFO, retratados pelos autores como os piores algorítimos de desempenho possível. Os testes realizados para incumbir essa alegação dos autores foi mostrar por meio do mapeamento aleatório, tento observado que em várias situações ele foi o mais eficaz quando o SO, precisava realizar mutilas escritas na cache, acarretando o (SWAP), no qual o valor removido seria reutilizado em um momento próximo. Dando o enfoque que o artigo [2], foi escrito a vários anos, acarretando um avanço na tecnologia, e de como o SO requisita os dados, não posso tomar essa decisão de não testar os algorítimos, mesmo que os dois sejam ineficientes, quando a abrangência de requisição aumente, sendo que atualmente a cache está com uma capacidade muito maior que há anos atrás, desta forma analisando os algorítimos utilizando tecnologias atuais, tirando minhas próprias análises sobre os mesmos.

Já considerando um artigo mais novo [1], retrata que as políticas, convencionais de substituição, relacionadas em LRU, não se caracterizam por ser as mais eficazes, no caso de sistemas de memória híbridos. Com isso, o artigo defende a criação de uma nova métrica de desempenho, chamada (AMAT), na qual se baseia em um intervalo de referência baseada em LRU.

Após vários testes feitos pelos autores [1], foi possível notar que os experimentos com base no índice de desempenho MALRU , bem como nos algorítimos LRU e HAP , mostra que o LRU obteve resultados ruins comparados com o HAP. Baseando nesses dados já consigo perceber que o LRU e FIFO não são bons algorítimos, mas como o ambiente utilizado não dá outros algorítimos além dos citados, irei demonstrar uma forma na qual um supere o outro.

III. METODOLOGIA

Objetivando analisar todas as medidas de localidades e algorítimos de substituição presentes na cache, como não Memoria-Virtual, foram utilizadas as seguintes propostas de arquiteturas no simulador-Amnésia.

Essa parte sera dividida em dois tópicos, no qual o primeiro irá tratar somente da arquitetura da cache, e o segundo com a arquitetura da Memória-Virtual.

A. Cache

Tabela I
Arquitetura Associatividade FIFO

TMC	L	B	ASS	TMR	AL
4KB	1	1	1v-8v	16KB	FIFO
8KB	1	1	1v-8v	16KB	FIFO
16KB	1	1	1v-8v	16KB	FIFO
32KB	1	1	1v-8v	32KB	FIFO
64KB	1	1	1v-8v	64KB	FIFO

TMC: Tamanho memoria Cache
L: tamanho da Linha
B: Tamanho do Bloco
ASS: Associatividade
TMR: Tamanho memoria Principal
AL: Algorítimo de substituição

A utilização da arquiterura I,o objetivo foi demonstrar todos os tipos de associatividade até o nível de oito, para buscar a localidade temporal, em um tamanho de cache de no máximo 64KB, com o algorítimo de substituição FIFO.

Tabela II
Arquitetura Associatividade LRU

TMC	L	B	ASS	TMR	AL
4KB	1	1	1v-8v	16KB	LRU
8KB	1	1	1v-8v	16KB	LRU
16KB	1	1	1v-8v	16KB	LRU
32KB	1	1	1v-8v	32KB	LRU
64KB	1	1	1v-8v	64KB	LRU

TMC: Tamanho memoria Cache
L: tamanho da Linha
B: Tamanho do Bloco
ASS: Associatividade
TMR: Tamanho memoria Principal
AL: Algorítimo de substituição

Artigo também é, definir qual dos dois algorítimos de substituição seria o melhor, com a arquitetura II a grande mudança foi nesse algorítimo de substituição de FIFO para LRU , com isso buscando a resposta de qual sera a melhor.

Tabela III Variação Tamanho da Liha FIFO

TMC	L	B	ASS	TMR	AL
4KB	1	1	1	16KB	FIFO
8KB	2	2	1	16KB	FIFO
16KB	4	4	1	16KB	FIFO
32KB	8	8	1	32KB	FIFO

TMC: Tamanho memoria Cache
L: tamanho da Linha
B: Tamanho do Bloco
ASS: Associatividade
TMR: Tamanho memoria Principal
AL: Algorítimo de substituição

Arquitetura III, busquei analisar a localidade Espacial, e nessa arquitetura o principal ponto, foi a mudança gradativa nos números de linhas, buscando aumentar o tamanho do bloco que pode ser colocado num mesmo conjunto.

Tabela IV
Variação Tamanho da Liha FIFO

TMC	L	B	ASS	TMR	AL
4KB	1	1	1	16KB	LRU
8KB	2	2	1	16KB	LRU
16KB	4	4	1	16KB	LRU
32KB	8	8	1	32KB	LRU

TMC: Tamanho memoria Cache
L: tamanho da Linha
B: Tamanho do Bloco
ASS: Associatividade
TMR: Tamanho memoria Principal
AL: Algorítimo de substituição

Como proposto na arquitetura III, com o intuito de analisar a localidade Espacial, mas com outro algorítimo de substituição, nessa nova arquitetura IV, será feito os mesmos passos da anterior, SO que mudando o algorítimo.

Tabela V
Trace

Tace	operacao	local de memoria
0	leitura de dados	0-F
1	gravacao de dafos	0-F
2	busca de instrucao	0-F

O trace V, retratar uma maneira na qual seja bom e ruim simultaneamente, para cada localidade, buscando o mais realista possível nas trocas de instruções do dia a dia, desta forma o trace utilizado tem 200 linhas nas quais variam entre instruções entre 0-2 , com os respectivos locais de memória de 0-F (Em Hexadecial).

B. Memoria-Virtual

Memoria-Virtual é um mecanismo simples para tradução de endereços virtuais em reais e para gerenciamento do espaço de memória. Com isso será demonstrado uma série de arquiteturas

com ou sem o TLB , para medir a quantidade de paga fault e o tempo de acesso da memória.

Tabela VI
Variação Tamanho da página

DM	PS	B	TLB	TMR	AL
16KB	2	1	nao	32KB	FIFO
16KB	4	1	nao	32KB	FIFO
16KB	8	1	nao	32KB	FIFO
16KB	16	1	nao	32KB	FIFO

DM : Tamanho do disco
PS: tamanho da página
B: Tamanho do Bloco
TMR: Tamanho memoria Principal
TLB:Translation Look-Aside Buffer AL: Algorítimo de substituição

Arquitetura VI, ainda sem a utilização do TLB , aumentei o tamanho da página, para verificar qual seria a nova ocorrência relacionada ao número de pega fault , além de testar o algorítimo FIFO nessa situação.

Tabela VII
Variação Com TLB

DM	PS	B	TLB	TMR	AL
16KB	2	1	nao	32KB	LRU
16KB	4	1	nao	32KB	LRU
16KB	8	1	nao	32KB	LRU
16KB	16	1	nao	32KB	LRU

DM : Tamanho do disco
PS: tamanho da pagina
B: Tamanho do Bloco
TMR: Tamanho memoria Principal
TLB:Translation Look-Aside Buffer AL: Algorítimo de substituição

Além do FIFO ,testarei com os mesmos padrões de arquiteturas VII o algorítimo LRU , para obter a informação de qual será o melhor.

Como o intuito primordial, das memórias e tentar nunca ler do disco e otimizar a velocidade, a utilização da TLB, será posta em prática nas próximas arquiteturas.

Tabela VIII
Utilização do TLB na Memória

DM	PS	B	TLB	TMR	AL
16KB	2	1	Sim	32KB	FIFO
16KB	4	1	Sim	32KB	FIFO
16KB	8	1	Sim	32KB	FIFO
16KB	16	1	Sim	32KB	FIFO

DM : Tamanho do disco
PS: tamanho da página
B: Tamanho do Bloco
TMR: Tamanho memoria Principal
TLB:Translation Look-Aside Buffer AL: Algorítimo de substituição

Tabela IX TLB

T Memoria	ALgoritimo
2KB	FIFO
4KB	FIFO
16KB	FIFO
32KB	FIFO

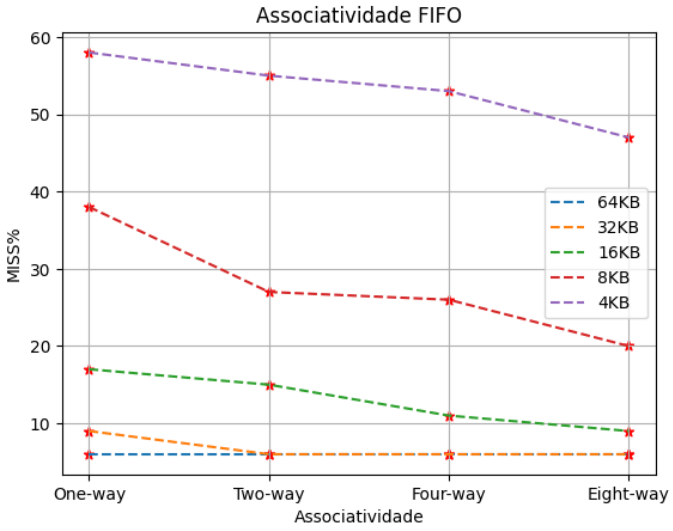
Portanto, realizarei os testes da arquitetura VIII, mas usando a TLB, para verificar qual terá um desempenho melhor.

IV. Avaliação dos Resultado

Nos gráficos subsequentes, há uma demonstração de como a memória cache se comporta ao ser usada com um grande conjunto de instruções de *load* e *store* (maior que sua capacidade) em cenários propostos pelas arquiteturas. Os gráficos, serão distribuídos em duas seções, nas quais serão, memória Cache e memória-virtual, nessa sequência.

A. Cache

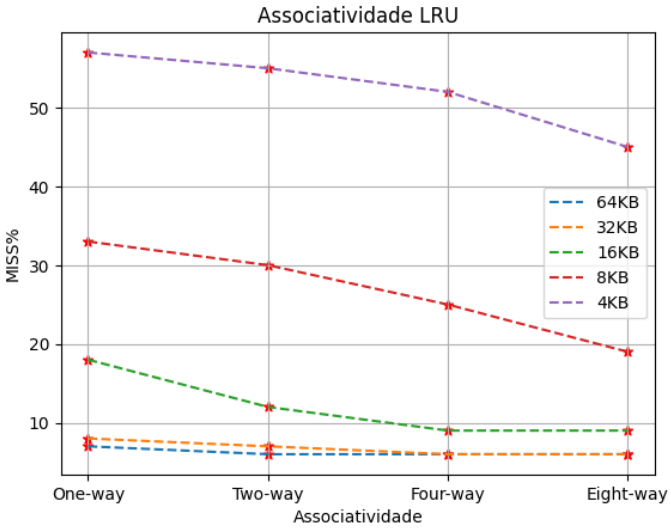
Gráfico I
Impacto da Associatividade FIFO



Ao analisar esse gráfico I , e possível perceber o quanto a associatividade, influenciou em diminuir a quantidade de chace miss, pelo fato de que aumentando a cache em si, em mais vias. Irei conseguir colocar mais palavras na cache e com isso não precisando ir até na memória principal requisitar os dados. Além disso, mesmo uma cache bem pequena como de 4KB,utilizando os princípios de localidade temporal, no qual visa que após a utilização de um dado a cache dele ser utilizado novamente será muito provável e com isso deixando de remover o dado da cache. Portanto, o algorítimo

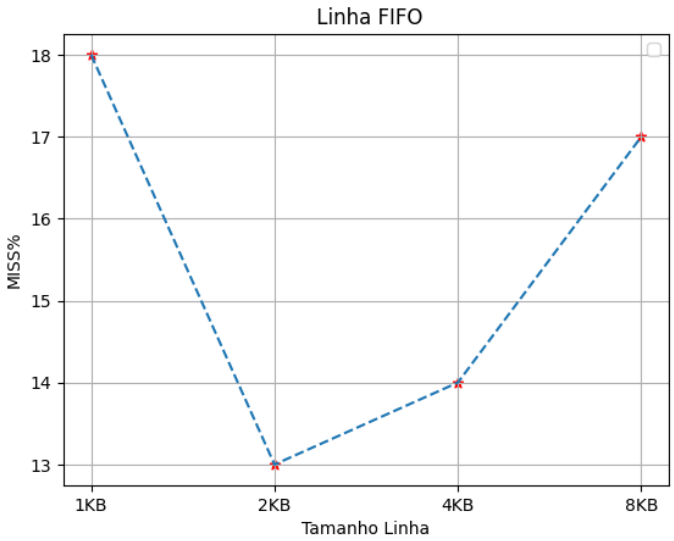
de substituição em caches maiores e com mais vias, é algo que não nos da noção tão profunda como em caches menores, mas no próximo gráfico a diferença entre os dois algorítimos em caches menores será mais vista.

Gráfico II
Impacto da Associatividade LRU



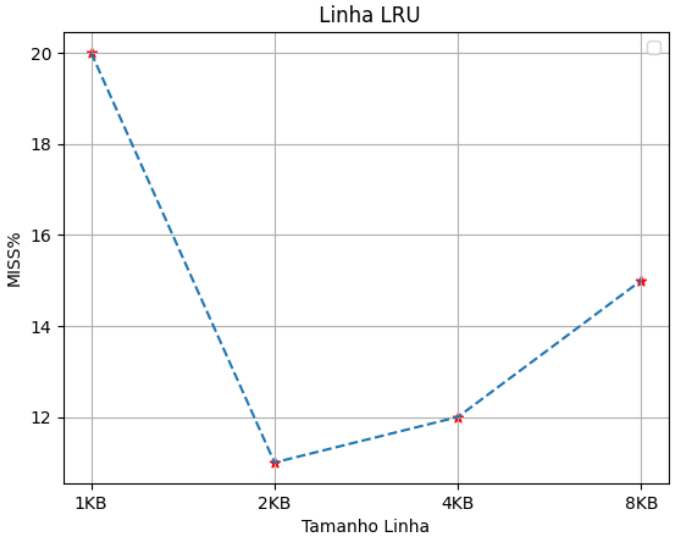
Fazendo a comparação desse gráfico II com o gráfico anterior, nota-se que o LRU teve uma queda de MISS suave, e em algumas vezes melhorou um pouco o MISS, mas como o LRU é um algorítimo mais difícil de ser implementado, comparado-se ao FIFO, obtendo essa pequena diferença, pode-se notar que o FIFO, mesmo tendo resultado piores que o do LRU, compensa mais na sua utilização pelo fato de ser mais simples e dar resultados muito próximos.

Gráfico III
Impacto da Linha FIFO



Quando foi testado no tocante ao aumenta das linhas no grafico III, utilizando a localidade espacial, visando a utilização dos dados mais proximos a ele, que podem ser utilizados, foi efetuado o enfio dos blocos de dados presentes na memoria principal para a cache. Mas a partir do gráfico, a utilização da localidade espacial de uma maneira equivocada pode ser terrível para a cache, pois uma linha maior significa o maior miss penalty, pelo motivo de demorar mais tempo para preencher a linha. Por conta do tamanho da linha ser maior que a cache, ocasionando um miss ratio maior.
Para ser verificado se o algorítimo de substituição poderia solucionar esse problema foi testado e obtido esse resultado.

Gráfico IV
Impacto da Linha LRU



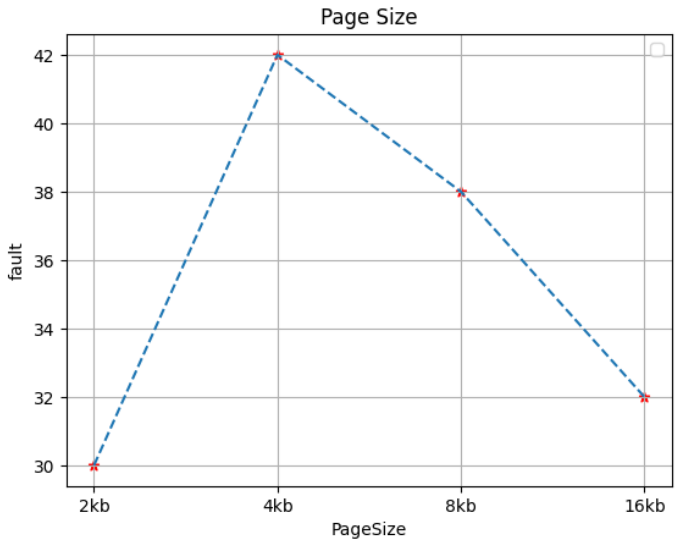
Mesmo utilizando outro algorítimo de substituição, ainda

tivemos um resultado inusitado IV em aumentar o tamanho da linha, sendo a unica coisa que pode garantir melhora, foi relacionada ao gráfico do FIFO onde teve uma linha assintoticamente maior que a do LR.
Portanto, nessa comparação, o LRU é um melhor algorítimo quando visamos aumentar o tamanho da linha.

B. Memória-Virtual

Com base no pressuposto que a memória virtual e uma técnica que nos permite ver a memória principal como uma cache de grande capacidade de armazenamento, foi feito testes sobre a mesma e com isso deu origem ao seguintes resultados:

Gráfico V

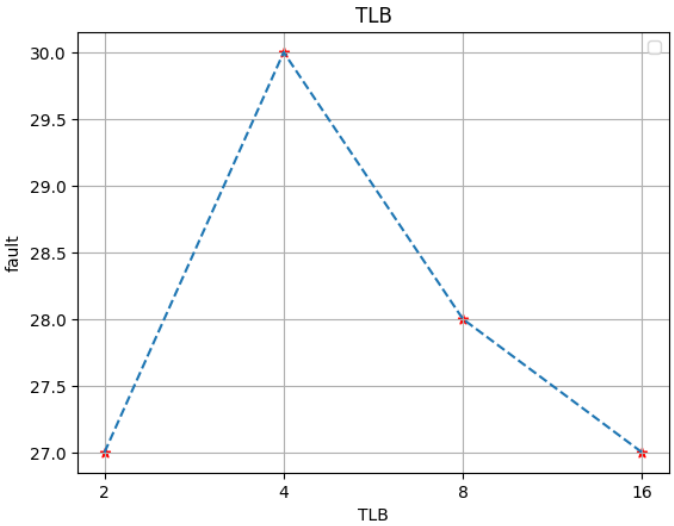


Como a paginação tradução de endereços virtuais em reais e para gerenciamento do espaço de memória, ela ajuda ter um tempo melhor de acesso, se tiver que procurar um tem na memória principa e no gráfico pode-se perceber que ao aumentar o tamanho da página de 2KB para 4KB grafico V , a quantidade de page fault aumentou em um nível chamativamente, sendo que ao procurar o motivo desse acontecimento, este está interligado ao trace pelo fato da página em alguns momentos estarem com o mesmo dado repetido.

Com isso há probabilidade de encontrar o endereço requisitado sobre sendo repetida, fazendo assim o gráfico subir, mas após aumentar gradativamente o tamanho da página a quantidade de page fault diminui.

Há utilização do TLB grafico VI que tem uma implementação em hardware, tendo sua utilização para ser mais rápido no ato de procurar o endereço na memória principal. Como podemos ver no gráfico,ocorreu o mesmo problema que na memória virtual, mas com uma melhoria acentuada. Sendo assim melhor que a pagina sózinha.

Gráfico VI



V. Conclusões

Após a realização dos testes em todos os cenários propostos, e uma análise dos gráficos apresentados, foi obtido resultados bastantes expressivos no quesito das caches, na qual nos testes das localidades, obtivemos as conclusões que, no ato de trocar a localidade, sem o conhecimento da mesma, pode ser muito danoso, no quesito tempo, ao encontrar um dado em memória. Conforme [2] , na qual os autores disseram que a diferença entre os desempenho dos dois algorítimos era muito pouca, foi possível demonstrar por meio dos gráficos essa comprovação, relatada pelos autores, quando mesmo que o desempenho do LRU sendo melhor, que do FIFO, a diferença obtida foi não relevante, fazendo assim que a implementação do LRU é inútil. Além dos aspectos da cache, também foi obtido dados relacionados a Memoria-Virtual, na qual mais uma vez a utilização dos algorítimos não mudaram de forma expressiva o resultado, tendo em si uma melhora quando utilizamos RLU, mas quando foi implementado o TBL a melhora em si, foi significativa, se baseando no tempo.

Concluindo, meu teste demonstra que os algorítimos RLU e FIFO, não tiveram resultados no quesito de otimização relevantes, mas os métodos de localidade espacial e temporal tiveram resultados alarmantes de utilização. E no quesito de Memória-Virtual o ato de adicionar o TLB seria algo fundamental para otimizar o tempo da memória.

VI. References

References

[1] Hai Jin ,Rentong Guo,Di Chen,Haikun Liu ,Xiaofei Liao,Miss Penalty Aware Cache Replacement for Hybrid Memory Systems, 12rd ed., vol. 39. COMPUTER-AIDED DESIGN, 2020.

[2] J. Smith and B. Goodman, "Instruction Cache Replacement Policies and Organizations" in *IEEE Transactions on Computers*, vol. C-34, no.3, pp. 234–241, March 1985.