KSH: Simulador de memória cache com carregamento dinâmico de módulos e execução de script de configuração

Alessandro Ribeiro da Silva[1] Ramon Duarte Pinho[2]
Pontifícia Universidade Católica de Minas Gerais
Av. Dom José Gaspar, 500 – Coração Eucarístico – CEP 30535-610
Belo Horizonte – Minas Gerais
{spdoido@hotmail.com}[1]-{ramon-duarte-pinho@bol.com.br}[2]

Resumo

Este trabalho apresenta um software para simulação de memória cache desenvolvido numa disciplina de graduação. Seu propósito é de servir como ferramenta didática para análise e aprendizado de diferentes estruturas de memória cache.

1. Introdução

Memória cache é um dos tópicos estudado na disciplina de Arquitetura de computadores II do curso de graduação de Ciência da Computação na PUC-MG.

Os exercícios de memória cache apresentam uma dificuldade geral para o estudante pois são extensos, trabalhosos e demorados. Para facilitar a execução desse tipo de exercício é interessante o uso dos simuladores, agilizando a análise comparativa e seus cálculos de dados estatísticos.

Tivemos contato com alguns simuladores de memória cache dos quais observamos os pontos positivos e negativos de seus aspectos didáticos.

Um dos simuladores analisados foi desenvolvido na mesma disciplina há um ano atrás por um grupo de alunas da PUC cujo artigo foi publicado no WSCAD 2002. O simulador delas limitava-se a um único tipo de arquitetura de memória cache, e a exibição de uma simulação íntegra e um resultado técnico. O simulador delas não apresentou um nome com o qual poderíamos nos referenciar[1].

Outro simulador analisado foi o Xcache32, que nos ofereceu muitas idéias de implementações pelos recursos de interface de usuário que este ofereceu[2].

Após uma análise deles definimos um conjunto de características ideais voltadas para o ensino que um simulador deveria possuir. Propomos o desenvolvimento de um simulador com alguns desses aspectos. Dentre esses aspectos desejávamos em nossa ferramenta: flexibilidade, facilidade de uso, interface amigável com o usuário, tutorial, e sem barreiras de línguas.

Uma das atividades desta disciplina deste semestre foi a criação de um simulador de memória cache que deveria mostrar o conteúdo das memórias, informar sobre os acessos à memória, simular 3 tipos de arquitetura de cache, simular as políticas de substituição FIFO e LRU[3].

Pensando na contribuição de nosso projeto, desenvolvemos um simulador com todas as características básicas necessárias e acrescentamos ao simulador uma carga dinâmica de módulos de arquitetura explicada no tópico 2.2. e um processador de script para torná-lo personalizável, explicada no tópico 2.1. Nosso simulador foi implementado na linguagem Java® na versão 1.4, editado pelo JCreator versão 2.00 e demos a ele o nome de KSH.

2. A ferramenta

Nosso simulador pode ser executado atualmente em duas interfaces que podem ser escolhidas pelo usuário: modo gráfico (GUI)[3] e modo console (tipo dos). Ambas as interfaces oferecem ao usuário as mesmas opções, dentre elas, abrir um arquivo de seqüência de endereços, entrar com um endereço e escolher uma configuração para as memórias. O modo gráfico apresenta suas vantagens, é mais amigável e os dados não devem ser inseridos numa ordem pré-determinada.

Figura 1. Imagem do simulador no modo gráfico para entrada de dados via teclado

Para operar o simulador o usuário deve configurar inicialmente as características das memórias. Por enquanto pode ser configurado para operar com as três arquiteturas básicas (*Associativo por Conjunto,*

Completamente Associativo, Mapeamento Direto), duas políticas de substituição (*FIFO, LRU*), duas técnicas de *escrita (Write Back, Write Through*), e o tipo de acesso a memória (*Paralelo, Sequencial*), estudadas na disciplina. O usuário também pode definir o tamanho da memória principal, o tamanho da memória cache, o tamanho do bloco, o tamanho da palavra e o tempo de acesso às memórias.

A execução pode ser feita passo a passo ou integral a partir de um arquivo de entrada com uma seqüência de endereços (*memory trace*), ou através da entrada dos endereços via teclado. O programa irá exibir o que ocorreu durante o acesso considerando acerto ou falta, leitura ou escrita, substituição de bloco e escrita na memória.

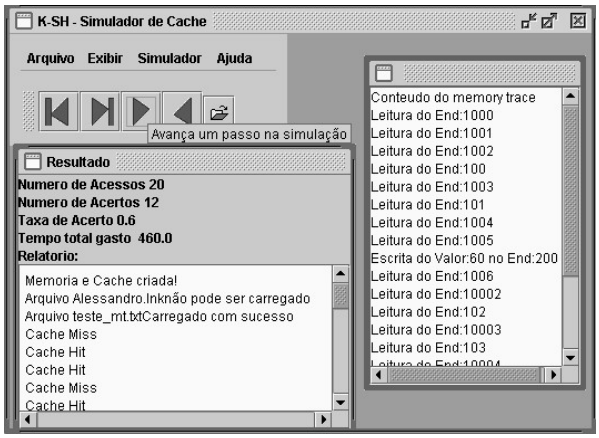

Figura 2. Foto da interface gráfica executando leitura de uma arquivo de seqüência de endereços.

Uma das características de implementação do nosso software foi a idéia de carregar dinamicamente alguns módulos permitindo que algumas características sejam independente do programa principal. O carregamento dinâmico foi criado com a finalidade de tornar o simulador flexível e expansível e será explicado com mais detalhes no tópico 2.2.

O processador de *Script* oferece ao usuário uma opção de automatização e personalização de algumas funções do simulador. Atualmente existe apenas um comando de inferência no simulador que define a configuração das memórias do simulador. Contudo a idéia é permitir que o arquivo de *script* crie uma verdadeira interação com o simulador. O *script* seria interessante didaticamente, pois uma seqüência de instruções poderiam ser repetidas numa ordem com padrões diferentes. Isso também seria utilizado para criação de *tutoriais*.

2.1. O Processador de Script

Este processador de script originou de um projeto da disciplina Linguagens de Programação, concluído no início do semestre de 2003. Supomos que um recurso de execução de Script em um simulador teria uma boa repercussão e por isso o adaptamos ao simulador Ksh.

Foi acrescentado somente um comando, que permite a configuração das memórias tornando o arquivo de script um arquivo de configuração.

Segue um exemplo de Script de Configuração.

<arquivo teste_script0.txt>

```
"proc main()
local TamMem, TamPalavra,
    identificadorArquitetura, Politica, Slots,
    TamBloco, Esc, Nways, TipoDeAcesso,
    TempoDeAcessoCache,
    TempoDeAcessoMemPrincipal, Str;
/****Iniciando configuracoes da cachê*****/
TamMem:=30000;/*Quantos Enderecos*/
TamPalavra:=2;/*Em bits*/
identificadorArquitetura:=1;/*
0 = Assoc por Conjunto/1 = Completamente Assoc
            2 = Mapeamento Direto*/
Politica:=0;/*    0 = FIFO/1 = LRU*/
Slots:=4;
TamBloco:=4;/*Quantas Palavras*/
Esc:=1;/*0 = WB/1 = WT*/
Nways:=2;/*Somente AC*/
TipoDeAcesso:=1;/*1 = Seqüencial/0 = Paralelo*/
TempoDeAcessoCache:=5;/*em ns*/
TempoDeAcessoMemPrincipal:=50;/*em ns*/
/*******Gerando String de Entrada*******/
Str:=TamMem+","+TamPalavra+","+
    identificadorArquitetura+","+Politica+","+
    Slots+","+TamBloco+","+Esc+","+
    Nways+","+TipoDeAcesso+","+
    TempoDeAcessoCache+","+
    TempoDeAcessoMemPrincipal;
/*Comando de carregar Ativar configuração*/
AtivarModulos(Str);
Endp"
```

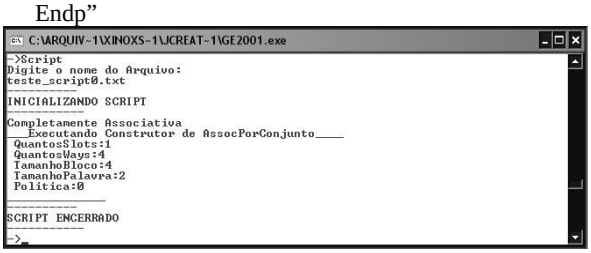

Figura 3. Script de teste executado.

2.1.1. Detalhes sobre o script. A linguagem do script possui comandos de definição de variáveis globais e locais, comandos de laços (for, while), comandos de atribuições de expressões a variáveis e um comando de condição (if ou if/else).

As expressões podem ser montadas utilizando os operadores +,-,*,/, onde o operador + quando se trabalha com strings é um símbolo de concatenação.

Os resultados das expressões podem ser comparados através dos operadores >(maior) ,<(menor) ,>=(maior ou igual) ,<= (menor ou igual) ,=(igual) ,<>(diferente).

Os resultados das comparações podem ser analisados como verdadeiro (numero diferente de zero ou String diferente de "") ou como falso através dos operadores and (e lógico), or (ou lógico) e not (negação).

O script possui uma função embutida para cálculo de raiz quadrada (sqrt), ainda dando suporte a

declaração de procedimentos, onde pode-se fazer um script semi-modular com possibilidade da utilização de recursividade.

È possível trabalhar com o console de Java a partir do script, para a entrada e saída de dados(System.in e System.out).

2.2. O Carregador Dinâmico

O carregador dinâmico foi implementado utilizando-se de alguns recursos do Java® permitindo que algumas classes sejam carregadas apenas quando requisitadas pelo programa. Apesar de que a atual estrutura do simulador ofereça o carregamento de apenas um tipo de módulo referente a arquitetura da memória cache, a idéia visa a instanciação de classes carregadas durante a execução do programa. Futuras implementações poderiam oferecer o carregamento de interfaces com o usuário.

A carga dinâmica pode ser implementada em outro momento e acrescentada à ferramenta de forma independente. Isso é uma característica de expansibilidade e no mais não limita o simulador com apenas sua capacidade atual.

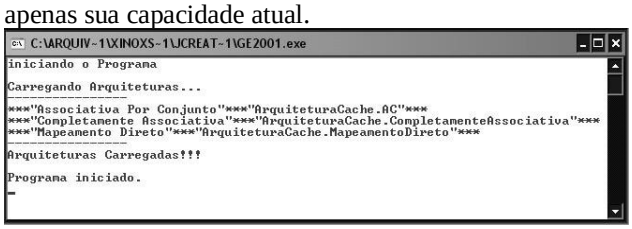

Figura 4. Estruturas sendo carregadas dinamicamente exibidas no modo console

A utilização deste recurso abrange todo o simulador, a carga de novas funções, novos padrões de comunicação, novas dimensões de execução e configurações de entrada e saída. Atualmente o carregamento dinâmico limita-se ao tipo de arquitetura que será utilizada na execução da simulação. A seguir, uma ilustração descritiva do funcionamento do carregador dinâmico implementado.

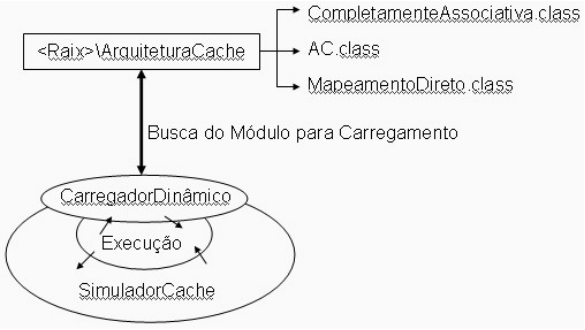

Figura 5. Diagrama do Carregador Dinâmico

Uma função do simulador permite a busca de todos os módulos das arquiteturas disponíveis num diretório, no caso ArquiteturaCache.

Após a fase de busca de módulos, o Carregador Dinâmico armazena uma referência para cada módulo válido juntamente com seu nome definido internamente no arquivo ".class", assim possibilitando uma instanciação futura.

Quando o usuário escolhe uma determinada arquitetura para a simulação, esta na verdade foi obtida através do Carregador Dinâmico, que fornece os identificadores(nomes internos) das classes que podem ser instanciadas.

Como seria difícil para o simulador ver a classe por seu identificador(nome interno), este é convertido para o identificador real de classe da linguagem Java e então instanciado.

Caso um usuário queira expandir seu simulador, é necessário que este possua somente o arquivo ".class" da arquitetura que deseja incrementar ao seu simulador.

Existe um arquivo molde de classe, na pasta de <Raiz do Simulador>\ArquiteturaCache, para facilitar ao programador a criação de novos módulos.

O programador do módulo deve implementar as operações básicas sobre uma cache como leitura e escrita. E avisar ao simulador quando ocorrer um acerto ou erro em alguma operação.

O módulo editado deve conter sua própria estrutura de armazenamento, assim não limitando uma possível grande expansão ou criação de mais de um nível de cache.

3. Resultados de Testes

Uma análise do resultado é feita a partir de alguns testes que preenchem duas tabelas com dados relativos à diversas configurações a partir da seqüência de endereços da figura 2. A execução pelo modo console consome mais linhas para apresentação do resultado, o modo gráfico é mais econômico.

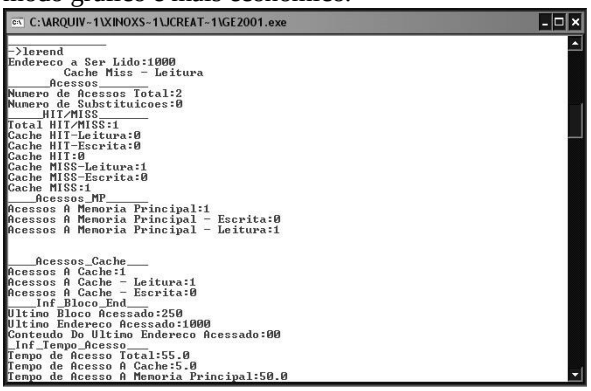

Figura 6. Exemplo de uma leitura feita no modo console.

Todas as configurações das simulações utilizadas aqui para teste possuem tamanho da palavra de 16bits, tamanho da memória principal de 16384 palavras, tempo de acesso a cache de 5ns, e tempo de acesso a memória principal de 50ns.

Sigla	Significado
CF	Configuração
TM	Tamanho da Cache (N.º de Slots)
TC	Tipo de Cache
PolS	Política de Substituição (LRU, FIFO)

NW	Número de conjuntos
TE	Técnica de Escrita
TA	Tipo de Acesso
MD	Mapeamento Direto
CA	Completamente Associativo
AC	Associativo por Conjunto
Prll	Paralelo
Seq	Seqüencial
WB	Write Back
WT	Write Through
X	Não importa qual escolha

Tabela 1.Siglas utilizadas na tabela 2.

Em caso de um cache miss de escrita, é escrito o conteúdo em questão diretamente na memória principal, então, para a seqüência de acessos analisadas foram colocados blocos com tamanho de 256 palavras, para forçar um cache hit de escrita.

CF	TM	TC	PolS	NW	TE	TA
(I)	4	MD	X	X	WT	Seq
(II)	4	MD	X	X	WT	Prll
(III)	4	MD	X	X	WB	Seq
(IV)	4	MD	X	X	WB	Prll
(V)	4	CA	X	X	WT	Seq
(VI)	4	CA	X	X	WT	Prll
(VII)	4	CA	X	X	WB	Seq
(VIII)	4	CA	X	X	WB	Prll
(IX)	4	AC	X	2	WT	Prll
(X)	4	AC	X	2	WB	Prll
(XI)	2	CA	FIFO	X	WB	Prll
(XII)	2	CA	LRU	X	WB	Prll

Tabela 2.Configurações de simulações.

Config.	N.º Acesso.	N.º Acertos	Taxa Acerto	Tempo (ns)
(I)	20	16	0.8	400
(II)	20	16	0.8	370
(III)	20	16	0.8	300
(IV)	20	16	0.8	280
(V)	20	17	0.85	350
(VI)	20	17	0.85	325
(VII)	20	17	0.85	250
(VIII)	20	17	0.85	235
(IX)	20	17	0.85	325
(X)	20	17	0.85	235
(XI)	20	15	0.75	375
(XII)	20	14	0.7	470

Tabela 3. Resultados de Simulações com as configurações da tabela 2.

4. Conclusão

Atingimos o objetivo de desenvolver um simulador com as características mínimas exigidas pelo professor. Acrescentamos algumas idéias discutidas durante o semestre baseado em análises de outros simuladores como o da Alexandra e o XCache32. Esse projeto nos deixou idéias para sua expansão e melhoramento.

Não foi possível concluir esse simulador com todas as características que havíamos proposto inicialmente. Ocupamos muito tempo com detalhes de implementação o que nos ensina que uma ferramenta pronta seria mais interessante ao estudante de memória cache. Apesar disso desenvolvemos uma ferramenta com dois aspectos interessantes que permite sua flexibilidade e uma configuração personalizada.

O estudo sobre memória cache foi eficaz, porém o tempo consumido durante seu desenvolvimento mostrou sua ineficiência. A partir daí estamos empenhados em desenvolver uma ferramenta com as características ideais para uso de futuros estudantes.

O carregador de módulos dinâmicos favorecido pela estrutura da linguagem Java® e o processador de scripts são nossa contribuição para futuras ferramentas didáticas visando flexibilidade, portabilidade e expansibilidade. Futuras ferramentas devem buscar uma interface amigável com o usuário e os módulos dinâmicos permitem a tradução do software com grande facilidade, vencendo barreiras nacionais e educacionais. Esse software ainda poderá apresentar uma versão que será executada a partir de um browser permitindo o fácil acesso a qualquer usuário do mundo.

Por meio de scripts será possível a criação de tutoriais para a ferramenta, embora a versão atual ainda não suporta esses recursos. Nosso plano sempre foi criar uma ferramenta didática, e por ser didática ela deve ser auto-instrutiva.

Nosso simulador ainda tem muito o que crescer. Faça um download do KSH pelo link: http://www.amonlord.hpg.com.br/arquivos/ksh.zip.

5. Referências

[1] Alexandra Silveira Costa, Christiane Vilaça Pousa, Carlos Agusto P. S. Martins; Simulador das meninas - "Projeto e desenvolvimento de um simulador de memória cache: análise funcional e de desempenho": http://www.simuladormemoriacache.kit.net/memoria_cache/artigo-memoria_cache2.pdf

[2]Xcache32: http://www.eie.fceia.unr.edu.ar/ftp/arquitectura_de_computadoras/SIMULADORES/XCACHE/XCACHE32/

[3] PATTERSON, David A. Organização e projeto de computadores: A interface HARDWARE/SOFTWARE. LTC - Rio de Janeiro, 2000.