# Comparação de Arquiteturas Utilizando Superescalar Simulator

## Ajudando a identificar as variações de velocidade e desempenho para diversas arquiteturas multi-core

Guilherme de Andrade Moura
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica
Campus Coração Eucarístico
Belo Horizonte – MG, Brasil

Ricardo Xavier Sena
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica
Campus Coração Eucarístico
Belo Horizonte – MG, Brasil

***Abstract*— Compreender o que torna uma arquitetura viável, em comparação a uma outra diferente é parte fundamental do aprendizado da arquitetura de computadores. Esse trabalho tem como função comparar duas arquiteturas multi-core através de um arquivo de testes, utilizando um simulador superescalar.**

***Keywords*— *Arquitetura de Computadores, Superescalar Simulator, Multi-Core.***

## I. INTRODUÇÃO

Desde o surgimento das arquiteturas de múltiplos núcleos, novas tecnologias e métodos para sua implementação são criadas de tempos em tempos. Seu proposito é tentar resolver um problema bem simples e muito trabalho dentro da computação em geral. Executar programas rapidamente com o mínimo de custo possível. Este artigo não tem como função propor um sistema de arquitetura ambiciosa que será mais rápido que as arquiteturas comuns. O objetivo que se tem através deste é elucidar a comparação entre duas arquiteturas. Queremos isto para ajudar a facilitar o entendimento e aprendizado de iniciantes. Para isto, resolvemos utilizar de uma ferramenta criada e idealizada por pessoas que possuem a mesma visão. Ao pesquisar mais sobre o Superescalar Simulator descobrimos que os desenvolvedores responsáveis por esta, a criaram na tentativa de facilitar o entendimento das funcionalidades dos simuladores de arquiteturas [1].

## II. FERRAMENTAS

### A. Objetivo

Na proposta oferecida pelo professor, foi decidido pela dupla realizar propor uma arquitetura e compara-la com outra arquitetura. Para isto utilizaremos o Superescalar Simulator fornecido pelo grupo I-Acoma da Universidade de Illinois, nos Estados Unidos. A arquitetura que usaremos para fazer o benchmarking junto a nossa será a que foi oferecida pelo professor para testes do simulador. Esta, veio anexada junto com os materiais do trabalho e teve seu funcionamento explicado por Matheus Alcântara Souza através de uma série de documentos [2]. Estes por sua vez foram-nos fornecidos pelo professor, Henrique de Cota Freitas, através do Sistema de Gerenciamento Acadêmico da Pontifícia Universidade Católica.

### B. Hardware

Todos os testes foram realizados em uma máquina com processador AMD Phenom (TM) II 3.20GHz com 8,00GB de memória RAM.

### C. Software

O Superescalar Simulator é um sistema nativo do Linux e a máquina que utilizamos possuí o Windows 10 Pro. Para evitar qualquer tipo de problema durante o desenvolvimento deste, foi decidido pelos autores não utilizar o Cygwin. Decidimos por instalar uma máquina virtual, mesmo que por este caminho acabemos comprometendo a velocidade de execução. O software de máquina virtual que utilizamos foi o Oracle VM Virtual Box. O sistema operacional que utilizamos na máquina virtual foi um Ubuntu 16.10 32 bits. A memória principal usada foi de 3916MB. Foi usado também aceleração VT-X, AMD-X, paginação aninhada e Paravirtualização KVM.

### D. Arquivo de Testes

Como mencionado na sessão A, nos foi fornecido uma série de arquivos e tutoriais para que pudéssemos desenvolver o trabalho da melhor forma possível. Dentre estes arquivos um arquivo de configuração “smp.conf” poderia ser utilizado para testar um outro arquivo “tt.in” no SESC. Pretendemos criar uma arquitetura que supere esse arquivo em questões de velocidade mantendo a qualidade do processamento

### E. Benchmark

O benchmark escolhido para fazer a mensuração dos dados de cada arquitetura foi o Crafty, um benchmark mais simples. Inicialmente consideramos em usar o benchmark Ocean do pacote SPLASH-2, pelo fato de que um dos membros do grupo já possuía experiência previa com este. Essa forma de benchmark trabalha com matrizes de uma forma em que suas operações em uma matriz quadrática[3]. Porém preferimos o

uso do Crafty pelo fato dele exigir menos esforço e também porque neste trabalho já nos foi fornecido um arquivo de testes e uma exemplificação de como funciona o SESC utilizando esta forma de benchmark, como mencionado nas sessões anteriores. O Crafty é um tipo de benchmark que trata o programa a ser processado como um jogo de xadrez. Esse bechmark faz o máximo para processar todos os possíveis movimentos em uma árvore. O tempo levado pelo benchmark para executar o programa de testes é de 24.99 milissegundos, 42753972 ciclos.

### *F. Arquitetura*

Como dito anteriormente, nossa arquitetura deve ser uma versão melhorada da arquitetura que nos foi oferecida para testar o simulador. Partindo desta informação, seria mais fácil e rápido se utilizarmos o arquivo de configurações que nos foi fornecido, e editá-lo de forma tentar melhora-lo. A partir de uma analise do arquivo da configuração e de como ele funciona, nós desenvolvemos uma arquitetura SMP (*Symmetric Multi-Processing*) com os seguintes detalhes:

TABELA I – DADOS DA ARQUITETURA ORIGINAL E DA NOVA ARQUITETURA

| | |
|---|---|
| **Quantidade de Núcleos:** | 32 |
| **Organização de Memória:** | Uma memória principal e uma cache |
| **Abordagem de Interconexão:** | Bus Entre L1 e L2 |
| **Memória L1:** | 1024 (WT) |
| **Memória L2:** | 1024 (WB) |
| **Mecanismo de Predição:** | Hibrido |

| | |
|---|---|
| **Quantidade de Núcleos:** | 128 |
| **Organização de Memória:** | Uma memória principal e uma cache |
| **Abordagem de Interconexão:** | Bus Entre L1 e L2 |
| **Memória L1:** | 2048 (WT) |
| **Memória L2:** | 1024 (WT) |
| **Mecanismo de Predição:** | Hibrido |

[a.] Detalhes da arquitetura proposta pela dupla

### *G. Dificuldades*

Inicialmente estávamos desenvolvendo um outro tipo de projeto que acabou se perdendo dado a um furto do computador que estávamos utilizando. Tivemos que adiar a apresentação da primeira etapa do projeto devido a esse acontecimento. Como perdemos grande parte do nosso antigo projeto decidimos recomeçar e realizar este, que ocuparia menos tempo e ainda assim teria certa relevância.

Já na fase dos testes enfrentamos outro problema, dessa vez com o nosso simulador. Durante as simulações apenas o resultado em relação ao cache-hit e cache miss era disponibilizado pelo terminal. O tempo era representado apenas por "#.##". Para descobrir a quantidade de tempo que cada teste de configuração levou, tivemos que analisar o arquivo resultado da construção. Os arquivos costumavam a ser incrivelmente extensos com grandes quantidades de linha, mas no inicio descobrimos que logo após os dados da arquitetura e passando um pouco uma sessão de calculo de clocks era possível identificar o tempo levado durante o processamento em conjunto com a quantidade de ciclos.

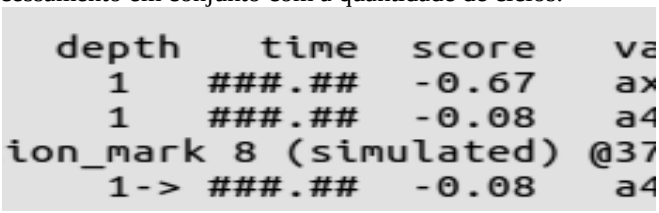



Imagem 1. Captura de tela editada demonstrando a falta de um resultado de tempo.

## III. DESENVOLVIMENTO

Para cumprir com o nosso proposito, devemos começar identificando quais as linhas de código do arquivo de configuração "smp.conf" correspondem as partes reais de um processador. Após compreender o funcionamento de cada linha deveríamos observar os números que as compreendiam e entender como eles afetam a latência e a qualidade de processamento. A partir dessas observações, tentaremos criar uma arquitetura que apresenta um speed-up em relação a original.

Na tentativa de descobrir fatores que poderiam nos proporcionar um speed-up decidimos realizar diversos testes de caráter experimental, onde fizemos pequenas alterações no arquivo smp.conf, em áreas em que poderiam apresentar-se alterações no desempenho. Depois de fazer estas alterações, realizamos o benchmark e comparamos a quantidade de tempo que o programa levou para realizar com o arquivo original.

### *A. Primeiro Teste*

Para o primeiro teste decidimos aumentar a quantidade de processos por nó. Para isto editamos uma das primeiras linhas do arquivo de configuração: "Procs Per Node". Inicialmente testamos a mudança de 32 processos para 128. A quantidade de tempo gasto acabou aumentando, porém, os ciclos permaneceram os mesmos. Não obtivemos o resultado desejado.

### *B. Segundo Teste*

Para o segundo teste aumentamos o tamanho da linha de cache. Editamos a terceira linha cacheLineSize. Assim como fizemos no primeiro teste, quadruplicamos o valor de 32 para 128. O tempo de execução diminuiu para 22.89 milissegundos

e os ciclos foram reduzidos para 29881955. Apesar de termos conseguido o speed-up que desejávamos, resolvemos não parar com os testes por aqui. Tomamos esta decisão para tentar identificar mais fatores que poderiam provocar um aumento na velocidade do processamento, visto que nosso trabalho preza por elucidar o funcionamento de simuladores.

### C. *Testes de Frequência*

Nos terceiro e quarto testes resolvemos alterar as linhas de frequência do arquivo de configuração. Para o terceiro alteramos a frequência do processador. No quarto, a frequência do Clock. Alteramos o valor das duas linhas de 5e9 para 10e9. Não observamos grandes alterações nos valores de tempo.

### D. *Testes de Associatividade*

Nos quinto e sexto testes resolvemos trabalhar com a associatividade das memórias L1 e L2. Inicialmente alteramos a politica de escrita da memória L2 de write-back para write-through. Ocorreu um pequeno speed-up de apenas 0.2 milissegundos. Inicialmente tentamos aumentar a associatividade de L2 apenas. Apesar do número de ciclos ter sido reduzido com o aumento, a quantidade de tempo aumentou em um segundo. Visto que este resultado ia contra nossa ideia inicial, decidimos reverter essa alteração. A alteração do valor foi de 8 para 16. Depois da reversão, aumentos a associatividade da memória L1. O aumento foi o mesmo que realizamos em L2. Apesar do tempo não ter apresentado variações, o número de ciclos caiu de 29881955 para 29871626. Decidimos manter essa alteração.

### E. *Últimos testes*

Nos últimos testes, resolvemos trabalhar inicialmente com alterações nos tamanhos das páginas. Primeiro diminuímos o tamanho de 4096 para 2048. Não obtivemos resultados pois o benchmark considerou o número de páginas que prescrevemos insuficiente. Decidimos reverter a alteração e aumentamos o tamanho da página para 8192. Não obtivemos mudanças significativas.

Por fim, decidimos aumentar o tamanho da memória L1. Dobramos o tamanho, de 1024 para 2048. Obtivemos um speed-up de 0.3 milissegundos. O número de ciclos caiu para 29865772. Dobrando novamente (para 4096), o tempo acabou aumentando em 1 milissegundo. Decidimos por manter o tamanho em 2048. Creditamos esse speed-up a uma combinação proporcional entre o aumento da associatividade e o aumento da memória L1. Decidimos por fim não realizar mais testes.

## IV. RESULTADO

Após várias alterações podemos dizer que conseguimos encontrar o resultado que queríamos. Obtivemos um speed-up e uma redução considerável de ciclos. Acreditamos não ter conseguido identificar e isolar todas as variáveis que podem provocar um speed-up, porém estamos satisfeitos por acreditarmos que nosso serviço pode ser utilizado para fins elucidativos no meio educacional. Inicialmente tínhamos a ideia de conseguir um Speed-Up assombroso, onde poderíamos reduzir o tempo do programa em até 50%, até nos deparamos com a realidade e percebermos que a realização de tal feito não é algo tão simples, porém estamos felizes com o que conseguimos. O a melhoria da velocidade foi de 9.89% e conseguimos reduzir os ciclos em 12888200.

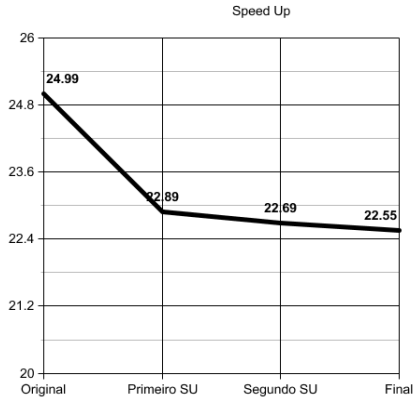
TABELA II - RESULTADOS

| **Tempo e Número de Ciclos** | | |
|---|---|---|
| ***Arquitetura*** | ***Tempo (mSecs)*** | ***Ciclos*** |
| Arquitetura Original | 24.99 | 42753972 |
| Primeiro Teste | 22.89 | 29881955 |
| Segundo Teste | 22.66 | 29871626 |
| Último Teste | 22.52 | 29865772 |

TABELA III – SPEED-UP

| ***Arquitetura*** | ***Tempo (mSecs)*** | ***Porcentagem*** | ***Melhoria(mSecs)*** |
|---|---|---|---|
| Arquitetura Original | 24.99 | 100% | 0 |
| Último Teste | 22.52 | 90.11% | 2.47 |

GRÁFICO I – SPEED-UP




## V. TRABALHOS FUTUROS

Nosso trabalho pode ser melhorado em diversas áreas. Quando começamos nossa ideia era criar algo que pudesse ser usado pelas áreas da educação de forma incentivar e facilitar o ensino de simuladores e de construção de arquiteturas. Apesar de termos identificado a presença de fatores que causaram uma melhoria na velocidade de processamento, ainda podemos ir mais longe. Podemos por exemplo identificar fatores e linhas de diversas configurações (não só SMP, mas CMP, AMD entre outras) que permitam criar um padrão básico que facilite ainda mais o ensino dessas simulações. Também poderíamos trabalhar essa ideia com outros simuladores e benchmarks diferentes.

## VI. Conclusão

Agradecemos pela oportunidade de trabalhar com esse sistema. Além de termos aprendido o funcionamento de uma nova ferramenta, o desenvolvimento da mesma facilitou a compreensão de várias facetas da matéria de arquitetura de computadores onde apenas na teoria não são fáceis de se assimilar. Ficamos felizes de ter conseguido chegar no resultado que esperávamos desde o surgimento dessa ideia e pelo de fato de que realizar as simulações foi um tanto quanto divertido, mesmo que nossos recursos e tempo fossem escassos.

Também gostaríamos de deixar aqui nosso agradecimento a todos os alunos da Pontifícia Universidade Católica que nos ajudaram com nosso trabalho e pelo suporte e paciência para nos ajudar a entender o funcionamento do simulador e de como configura-lo propriamente. Principalmente a monitora Ana Paula, bolsista de iniciação cientifica pela FAPEMIG e Gustavo Luiz que nos elucidou a respeito dos vários tipos de benchmark e de como alterar o arquivo de configurações. Também parabenizamos grande parte do corpo docente da PUC Minas por muitas vezes enfocar trabalhos como uma forma de ensinar pessoas que não possuem o mesmo acesso ao conhecimento nas áreas da computação que possuímos. Consideramos isto nobre e incentivamos esse tipo de comportamento.

Por fim, concluímos com nosso objetivo, aprendemos mais sobre simulações superescalares. Agradecemos novamente a chance de poder realizar esse trabalho.

## Referencias

Inicialmente tivemos muitos problemas ao começar a desenvolver o trabalho, tanto devido a forças externas quanto internas. Porém com o suporte oferecido tanto pelos alunos que já fizeram a matéria quanto pelo professor com o material conseguimos concluir esse trabalho e cumprir com nossos objetivos.

[1] DRAKOS, Nikos; MOORE, Ross; ORTEGO, Pablo Montensinos; SACK, Paul; “SESC: SuperESCalar Simulator”, 1999 <retirado de: http://iacoma.cs.uiuc.edu/~paulsack/sescdoc/ as 16:42 do dia 11 de Junho de 2017>

[2] SOUZA, Matheus Alcantâra; “Tutorial básico de instalação do SESC – Superescalar Simulator” <retirado do Sistema de Gêrenciamento Acâdemico da Pontifica Univerisdade Catolica de Minas Gerais as 10:00 do 3 de Junho de 2017>

[3] FONSECA, Gustavo; LUIZ, Gustavo; SENA, Ricardo; “Trabalho de Arquitetura de Computadores 3”, 2016 <fornecido por Gusatvo Luiz e Ricardo Sena em Maio de 2017>