Análise de desempenho do RadixSort em diferentes arquiteturas no simulador SESC

Gabriel O. Campos, Vinicius C. Pires, Brenon Henrique

Graduandos em Ciência da Computação

Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

Belo Horizonte - MG - Brasil

***Abstract*— With the need of better and faster solutions in software and hardwares, a SESC(Super Escalar Simulator) simulation was used to verify the best performace in architetures using the RadixSort algorithm, utilizing different type of architetures with different size of cores and cache configurations.**

Keywords— SESC, architetures, cache

***Resumo*— Com a necessidade de soluções de software e hardwares mais rápidas e eficazes foi utilizado o simulador SESC(Super Escalar Simulator) para verificar o melhor desempenho em arquiteturas no uso do algoritmo RadixSort, utilizando arquiteturas com diferentes tamanhos de núcleos e configurações de memória cache.**

Palavras-chave — SESC, arquiteturas, cache

I. Introdução

A cada dia aumentam o número de aplicações que exigem grande quantidade de processamento de dados, com isso surge a necessidade de programas que executam de forma mais rápida e eficiente. Para solucionar essa demanda são utilizados simulações para verificar o melhor tipo de arquitetura que pode ser utilizada em um problema..

O objetivo deste trabalho é analisar o melhor tipo de arquitetura para realização de problemas em RadixSort tendo em consideração variação no número de processadores e alteração na configuração de memória cache.

Configurou-se o simulador SESC para realizar testes utilizando diferentes combinações de tamanho de processadores com diferentes tipos de configurações de memória, com o objetivo de avaliar qual o melhor tipo de arquitetura. Para essas aplicações usamos o benchmark do RadixSort.

II. Simulador e Benchmark

A. Simulador SESC

O SESC é um simulador de arquiteturas superescalares capaz de alterar a propriedade de memória cache para avaliação da execução de vários tipos de benchmarks.

B. *RadixSort*

O benchmark utilizado foi o do RadixSort, o algoritmo RadixSort utilizado consiste em ordenar um vetor grande de números, O RadixSort ordena o vetor olhando a quantidade de dígitos no número,olhando sempre do dígito mais à direita do número até o dígito mais à esquerda.

C . Quantidade de processadores.

O tamanho dos processadores utilizadas na avaliação do desempenho foram de 1,2 e 4 núcleos, a fim de ter uma maior diversidade de resultados para melhor avaliação de desempenho.

D. Configurações da memória

Foram utilizadas simulações com memórias cache L1 e L2 de tamanho 32 Kbytes e 1 Mbyte respectivamente. Também foi testado com memórias cache do tipo compartilhadas e privadas.

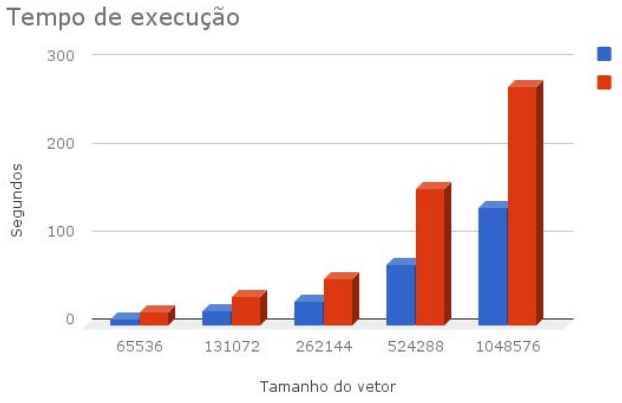
Memória Cache			Processadores		
Tamanho		Tipo			
L1	L2	As duas	Número		
32 Kb	1 Mb	Compartilhada	1	2	4
32 Kb	1 Mb	Privada	1	2	4

III. Execução

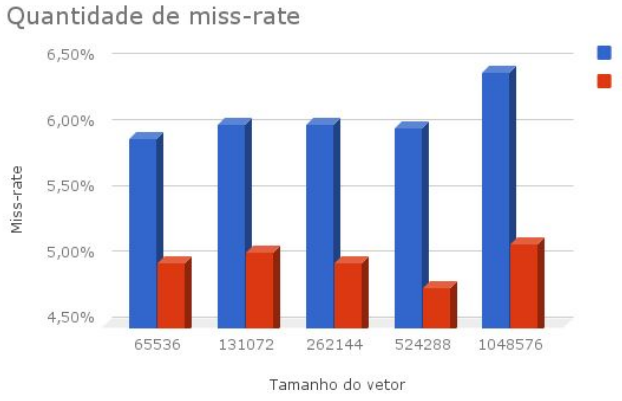
A seção de execução será dividida em quatro tópicos, primeiramente analisaremos a execução do algoritmo em arquiteturas de memória cache compartilhada e, em seguida, analisaremos o mesmo efeito para cache de memória privada. No terceiro tópico será abordado a comparação de desempenho obtidos pelas alterações feitas. Todos os testes destes tópicos foram executados utilizando para o mesmo benchmark, RadixSort.

A. Execução do algoritmo com 1 processador e memória cache compartilhada e privada

Dividiu-se a execução do algoritmo em tamanhos de arrays a serem ordenados. Na primeira execução foi alterado as configurações para 1 processador e memória cache compartilhada e privada. Como vemos na Figura 1,a execução desse primeiro teste revelou que com uma memória compartilhada existe um melhor tempo de execução e na Figura 2 a memória privada oferece uma menor quantidade de miss-rate



(Figura 1).

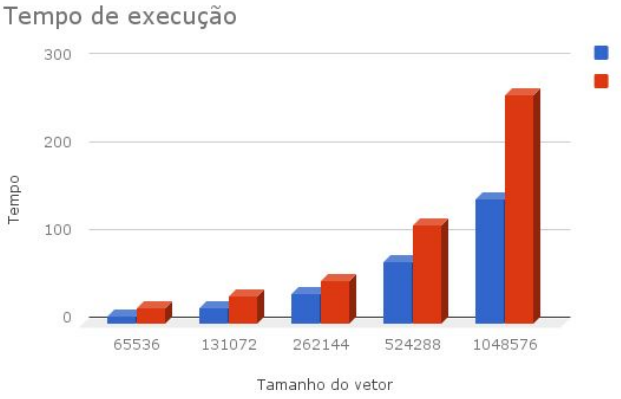


(Figura 2)

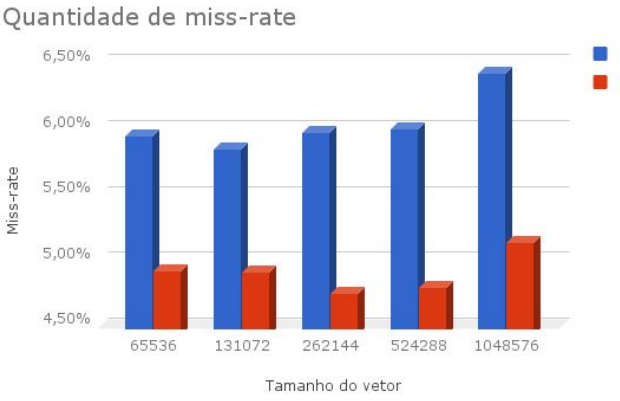
(Legenda: Coluna vermelha - Memória Compartilhada,Coluna Azul - Memória Privada)

B. Execução do algoritmo com 2 processadores e memória cache compartilhada e privada

Já na segunda execução foi alterado as configurações para 2 processadores com os mesmos tipos de configurações de memória. Como vemos na Figura 1,a execução desse teste mostrou que para com uma arquitetura de 2 processadores o mesmo resultado é encontrado, a memória cache compartilhada ganha em tempo de execução (Figura 3) e a privada ganha quando olhando a menor quantidade de perda de instruções(Figura 4)..



(Figura 3).



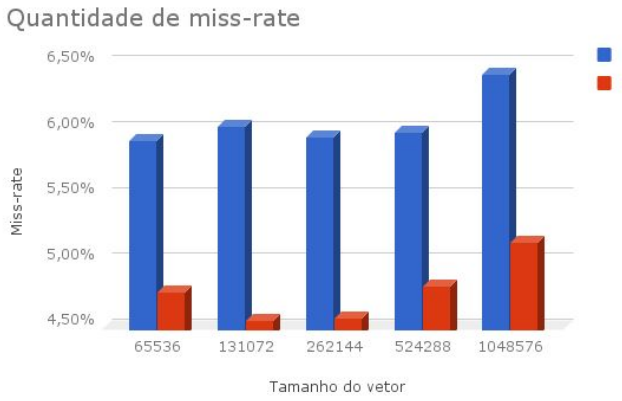
(Figura 4)

(Legenda: Coluna vermelha - Memória Compartilhada,Coluna Azul - Memória Privada)

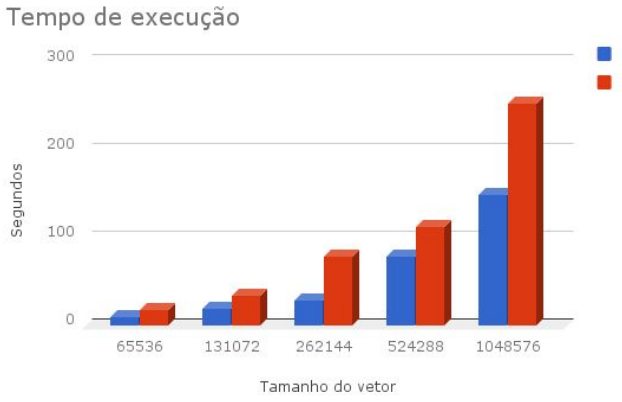
C. Execução do algoritmo com 2 processadores e memória cache compartilhada e privada

No terceiro teste foi executado o algoritmo agora com 4 processadores,mantendo o tamanho e as configurações de memória cache com compartilhado e privado, para verificar se o resultado para esse tipo de arquitetura manteria o mesmo.

Podemos ver melhor esses resultados com a (Figura 5) e (Figura 6) e em seguida analisaremos os resultados.



(Figura 5)



(Figura 6)
(Legenda: Coluna vermelha - Memória Compartilhada,Coluna Azul - Memória Privada)

D. Análise geral e outros benchmarks

Primeiramente, nota-se que ao comparar os tempos de execução das memórias compartilhadas e privadas,vemos claramente que a memória compartilhada gasta quase que metade do tempo na execução do algoritmo.

Observou-se também que para prevenir miss-rates é favorável a execução do algoritmo utilizando uma memória privada, pois a executou com aproximadamente 1% de miss-rates a menos que a memória compartilhada.

Uma tabela completa com os resultados para as execuções do benchmark com essas configurações realizadas está disponível no seguinte link: https://docs.google.com/spreadsheets/d/1l8l4GdaFlhAb2jK2otRVkJWm8MUtWlR4VMQOgNOoj1E/edit?usp=sharing

IV. Conclusão

Com a utilização do Simulador SESC conclui-se que na execução do algoritmo RadixSort é mais eficiente utilizar a memória cache compartilhada quando se necessita de maior velocidade de execução na resolução desse algoritmo. Porém, se preferir diminuir a quantidade de miss-rate é mais eficiente realizar a execução com memória cache privada

Como trabalhos futuros propomos a realização de benchmarks com diferentes tamanhos de memória cache Para verificação de um possível maior desempenho na execução desse problema. E baseando-se nos resultados determinar a relação ideal de quantidade de núcleos com tipo de memória cache e tamanho de memória cache.

Referências

[1] RadixSort. Disponível em : https://pt.wikipedia.org/wiki/Radix_sort

[2] Sesc Documentation. Disponível em: http://sesc.sourceforge.net//