UNICAMP - Universidade Estadual de Campinas

Instituto de Computação

MO601 - Arquitetura de Computadores II

Prof. Rodolfo Jardim de Azevedo

Projeto 4

Replicando resultados do artigo Memory Centric Characterization and Analysis of SPEC CPU2017 Suite

Fabrício Ferreira da Silva

RA 231900

Campinas – SP

Junho de 2023

Sumário

1 Objetivo

Desenvolver um experimento para replicar algum dos resultados do artigo Memory Centric Characterization and Analysis of SPEC CPU2017 Suite.

2 Especificação

Replique um gráfico do artigo, ou pedaço dele para o caso das figuras iniciais. Especificamente:

- Figuras 1, 2, 3 e 4: Escolher uma figura e replicar os dados de 3 programas apenas dessa figura.
- Figuras 5, 6, 7, 8, 9 e 10: Replicar um sub-gráfico apenas.

3 Ambiente de teste

Esse benchmark foi executado usando o seguinte ambiente:

- Model : Avell High Performance C62 LIV
- OS: Ubuntu 18.04.6 LTS 5.4.0-150-generic
- CPU: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
- L1CacheSize: 64KB per core, L2CacheSize : 256KB per core, L3CacheSize : 12MB
- MaxClockSpeed : 2592
- NumberOfCores : 6
- NumberOfLogicalProcessors: 12
- Memory: 2 x 04CB AM1P26KC8T1-BAAS 8 GB (8589934592) at 2667

4 SPEC CPU 2017 benchmark

O pacote de benchmark SPEC CPU® 2017 contém conjuntos intensivos de CPU da próxima geração, padronizados pelo setor, da SPEC para medir e comparar o desempenho intensivo de computação, enfatizando o processador, o subsistema de memória e o compilador de um sistema.

A SPEC projetou esses conjuntos para fornecer uma medida comparativa de desempenho de computação intensiva na mais ampla gama prática de hardware usando cargas de trabalho desenvolvidas a partir de aplicativos de usuários reais. Os benchmarks são fornecidos como código-fonte e requerem o uso de comandos do compilador, bem como outros comandos por meio de um shell ou janela de prompt de comando.

O pacote de benchmark SPEC CPU 2017 contém 43 benchmarks, organizados em quatro suítes:

As suítes SPECspeed® 2017 Integer e SPECspeed® 2017 Floating Point são usadas para comparar o tempo de um computador para concluir tarefas únicas.

As suítes SPECrate® 2017 Integer e SPECrate® 2017 Floating Point medem a produtividade ou o trabalho por unidade de tempo.

5 Gráfico escolhido para replicar

Foi escolhido para esse experimento o gráfico disponível na figura 1 do artigo, onde o objetivo é a contagem dinâmica de instruções para cada benchmark no pacote. Cada benchmark é dividido em workloads de entrada e apresentado na ordem de seus subconjuntos, com uma indicação da média do subconjunto no final. Esses resultados foram obtidos usando as instruções de evento de hardware do perf.

5.1 Programas escolhidos

Como o objetivo do experimento era replicar apenas 3 programas do gráfico, os programas escolhidos foram:

- 505.mcf_r
- 531.deepsjeng_r
- 605.mcf_s

5.1.1 505.mcf_r e 605.mcf_s

505.mcf_r e 605.mcf_s representam o mesmo programa, porém executados em subconjuntos diferentes, o 505.mcf_r pertence ao SPECrate e o 605.mcf_s pertence ao SPECspeed. É um benchmark derivado do MCF, um programa usado para agendamento de veículos de depósito único em transporte público em massa. O programa é escrito em C. A versão de benchmark usa quase exclusivamente aritmética inteira.

O programa é projetado para a solução de (sub-)problemas de programação de veículos de depósito único que ocorrem no processo de planejamento de empresas de transporte público. Considera um único depósito e uma frota de veículos homogênea. Com base em um plano de linhas e frequências de serviço, são derivadas as chamadas viagens programadas com locais e horários fixos de partida/chegada. Cada uma dessas viagens programadas deve ser atendida por exatamente um veículo. As ligações entre essas viagens são as chamadas viagens de cabeça morta. Além disso, há viagens pull-out e pull-in para sair e entrar no depósito.

5.1.2 531.deepsjeng_r

531.deepsjeng_r é baseado em Deep Sjeng WC2008, o Campeão Mundial de Xadrez de Velocidade de Computador de 2008. Deep Sjeng é uma reescrita do antigo programa Sjeng-Free, focado em obter a maior força de jogo possível.

Ele tenta encontrar o melhor movimento por meio de uma combinação de pesquisa de árvore alfa-beta, ordenação avançada de movimentos, avaliação de posição e poda heurística para frente. Na prática, explorará a árvore de variações resultantes de uma dada posição a uma dada profundidade de base, estendendo variações interessantes mas descartando as duvidosas ou irrelevantes. A partir desta árvore, a linha de jogo ideal para ambos os jogadores ("variação principal") é determinada, bem como uma pontuação que reflete o equilíbrio de poder entre os dois.

5.2 Execução do experimento

Acessar a pasta raiz do SPEC 2017 e executar o arquivo **shrc** que configura as variáveis de ambiente, em seguida para gerar os resultados foram usados os seguintes comandos:

- perf stat runcpu --config=gcc-try1 --size=all --iterations=2 505.mcf_r
- perf stat runcpu --config=gcc-try1 --size=all --iterations=2 531.deepsjeng_r
- perf stat runcpu --config=gcc-try1 --size=all --iterations=2 605.mcf_s

A flag “--config" define o arquivo de configuração que será usado na execução, esse arquivo é uma cópia da configuração original do próprio SPEC para o compilador GCC.

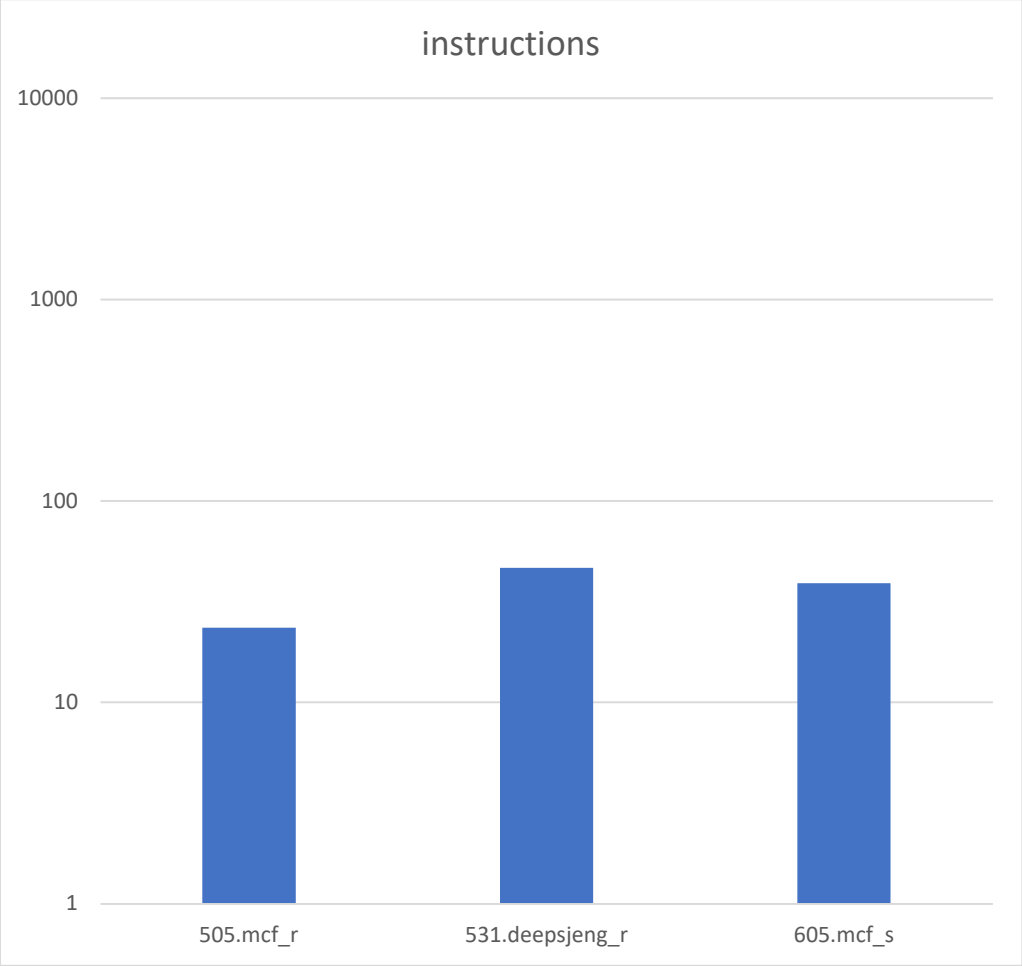
A flag “--size=all” define quais conjuntos de entrada serão usados, usando “all” serão executados todos eles sendo: test (para testar se os executáveis são funcionais), train (conjunto de entrada construído usando otimização direcionada por feedback e usado para binários de treinamento) e ref (conjunto de dados cronometrados dos aplicativos reais, que se destina a uma execução do tipo reportable).

Por fim a flag “--iterations=2” indica o número de iterações para cada conjunto de entrada, para a execução reportable funcionar corretamente são necessárias pelo menos duas iterações.

5.3 Resultados

Usando os comandos acima foram obtidos os seguintes resultados:

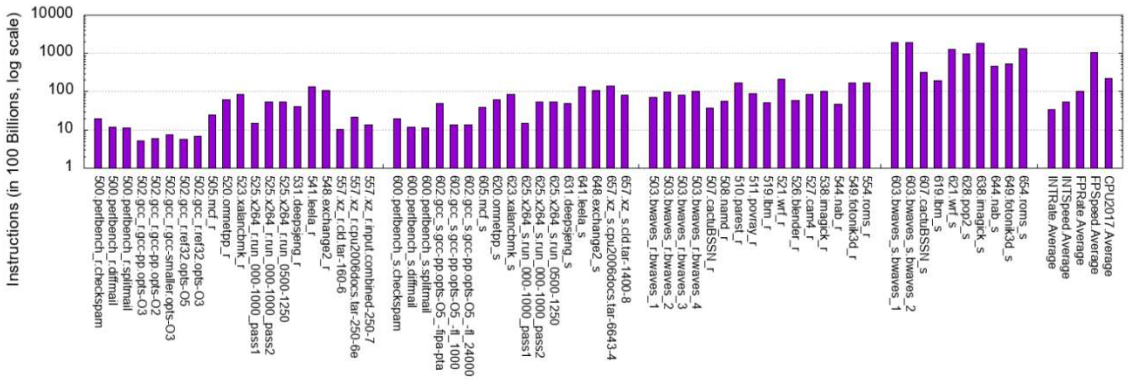
Resultados da execução do SPEC CPU 2017		
Suíte	**Programa**	**Instruções**
Intrate	505.mcf_r	2.349.059.412.358
Intrate	531.deepsjeng_r	4.656.551.718.039
Intspeed	605.mcf_s	3.899.739.665.271



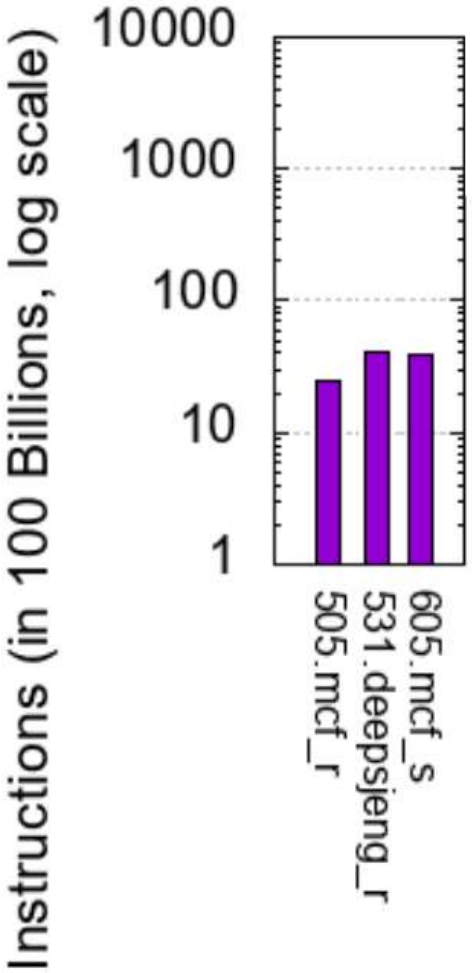
5.3.1 Comparação dos resultados com os presentes no artigo

Para comparar o número de instruções obtidas no meu computador com as métricas do artigo a primeira etapa foi coletar a figura 1 disponível no artigo e separar em mesma escala apenas aqueles que eu executei, para isso eu usei o **paint** e a ferramenta de captura do windows.

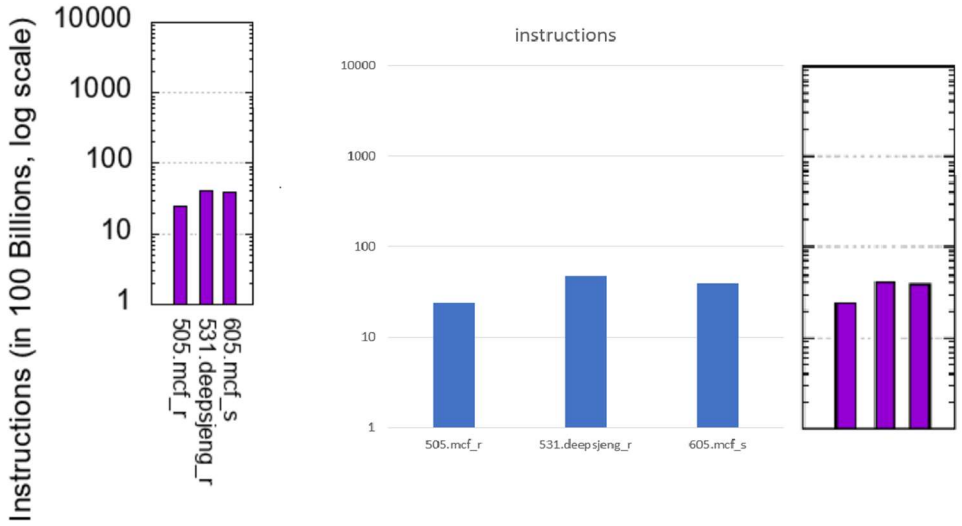
Figura 1 original do artigo



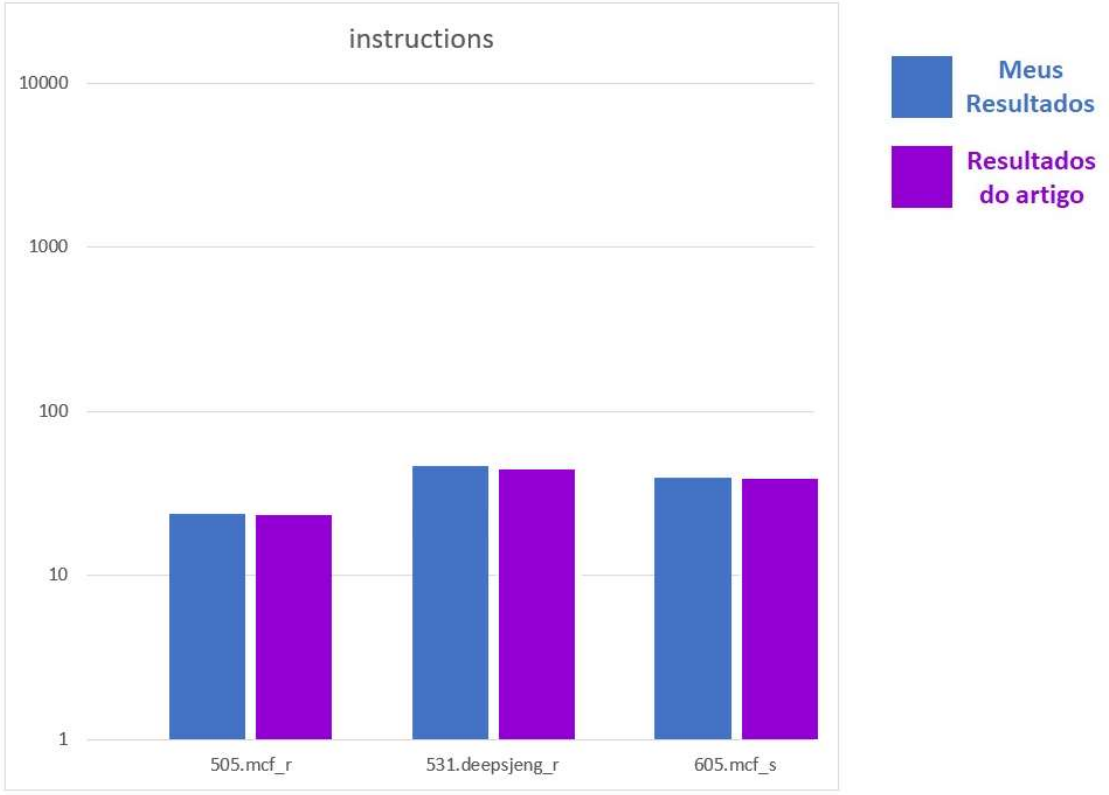
Recorte em mesma escala dos resultados de interesse



Em seguida também usando o **paint** eu redimensionei as figuras tanto do meu gráfico quanto do artigo para que ambas ficassem com as escalas em mesma proporção.



Por último com os gráficos em mesma dimensão também usando o **paint** eu fundi ambos os gráficos para criar o gráfico final para a comparação.



6 Conclusão

Nesse projeto para as especificações do meu computador, usando as configurações do comando **runcpu** descritas anteriormente e o compilador GCC, consegui resultados muito próximos dos presentes no artigo para os programas 505.mcf_r, 531.deepsjeng_r e 605.mcf_s.

A pequena variação pode ter sido causada por dois problemas, o primeiro deles é que já que a ferramenta usada para fundir os gráficos foi o **paint** e foi feito de forma totalmente manual pode ter sido que tenha ocorrido uma pequena deformação na dimensão dos gráficos, outro ponto é que como o artigo não especificou a versão do GCC usado, o número de instruções pode variar entre diferentes versões do mesmo compilador devido a mudanças na implementação do compilador. Essas mudanças podem ser feitas para melhorar o desempenho, corrigir bugs ou adicionar novos recursos. Além disso, diferentes versões do compilador podem ter diferentes níveis de otimização que podem afetar o número de instruções geradas pelo compilador.

Aprofundei ainda mais meus conhecimentos em ferramentas de benchmark além de conseguir aplicar os conhecimentos adquiridos no projeto 3 nesse experimento, o que o tornou sua implementação muito mais rápida.

Concluindo, ao fim desse projeto tive um aprofundamento no conhecimento do uso e funcionamento dos programas e configurações da ferramenta usada, o que torna possível que em projetos futuros onde eu precise utilizar o SPEC já terei uma ótima base para coletar os dados necessários.