|  |  |  |
| --- | --- | --- |
| Z:\Meu Drive\03. Doutorado\30. Doutorado IC-Unicamp\2022-2\Technical Report Template\logo-unicamp-name-line-blk-red-0120.png | **Universidade Estadual de Campinas**  **Instituto de Computação**  **Arquitetura de Computadores II – MO601**  **Prof. Rodolfo Jardim de Azevedo** | Z:\Meu Drive\03. Doutorado\30. Doutorado IC-Unicamp\2022-2\Technical Report Template\logo-ic-unicamp-slant-tint-beg-sky-ora-0120.png |

**Projeto 3**

**Experimentar ferramentas e coletar dados**

**Rubens de Castro Pereira**

**RA 217146**

Campinas – SP

Maio de 2023

Índice

[1 Introdução 4](#_Toc136418318)

[2 Ambiente de Experimentação 4](#_Toc136418319)

[3 Ferramentas experimentadas 5](#_Toc136418320)

[3.1 SPEC CPU 2017 benchmark 5](#_Toc136418321)

[3.1.1 Instalação e configuração 5](#_Toc136418322)

[3.1.2 Execução 6](#_Toc136418323)

[3.1.3 Resultados 6](#_Toc136418324)

[3.2 Simulador multi-core Sniper 8](#_Toc136418325)

[3.2.1 Instalação e configuração 8](#_Toc136418326)

[3.2.2 Execução 9](#_Toc136418327)

[3.2.3 Resultados 9](#_Toc136418328)

[3.3 Perf profiler 10](#_Toc136418329)

[3.3.1 Instalação e configuração 10](#_Toc136418330)

[3.3.2 Execução 10](#_Toc136418331)

[3.3.3 Resultados 11](#_Toc136418332)

[3.4 PARSEC Benchmark Suite 3.0 12](#_Toc136418333)

[3.4.1 Instalação e configuração 12](#_Toc136418334)

[3.4.2 Execução 13](#_Toc136418335)

[3.4.3 Resultados 13](#_Toc136418336)

[3.5 Rodinia benchmark 15](#_Toc136418337)

[3.5.1 Instalação e configuração 15](#_Toc136418338)

[3.5.2 Execução 15](#_Toc136418339)

[3.5.3 Resultados 16](#_Toc136418340)

[3.6 Intel Pin 17](#_Toc136418341)

[3.6.1 Instalação e configuração 17](#_Toc136418342)

[3.6.2 Execução 17](#_Toc136418343)

[3.6.3 Resultados 18](#_Toc136418344)

[3.7 Dinero cache simulator 19](#_Toc136418345)

[3.7.1 Instalação e configuração 19](#_Toc136418346)

[3.7.2 Execução 19](#_Toc136418347)

[3.7.3 Resultados 20](#_Toc136418348)

[4 Considerações sobre o aprendizado nesse projeto 24](#_Toc136418349)

[5 Apêndice 24](#_Toc136418350)

[5.1 PARSEC Benchmark Suite 3.0 \* 24](#_Toc136418351)

# Introdução

Esse trabalho tem o propósito de utilizar algumas ferramentas de avaliação de arquitetura de computadores com a coleta de dados da execução de *benchmarks* e programas que exploram aspectos como tempo de processamento, número de instruções executadas e uso de memória RAM e cache. As ferramentas utilizadas foram SPEC CPU 2017, simulador multi-core Sniper, Perf profiler, Parsec benchmark, Rodinia benchmark, Intel Pin e Dinero cache simulator.

Os resultados obtidos na execução das ferramentas estão organizados no repositório Github por meio do link <https://github.com/rubenscp/RCP-MO601-Project-03>.

A Seção 2 apresenta o ambiente de experimentação, a Seção 3 detalhada a execução e resultados alcançados em cada ferramenta, a Seção 4 descreve considerações sobre o aprendizado neste projeto e a Seção 5 apresentas as conclusões do trabalho.

# Ambiente de Experimentação

O computador utilizado nesse trabalho será denominado como “Laptop Rubens” e o sistema operacional base é o Windows 10 Pro 22H2, contudo para a execução de todas as ferramentas foi utilizado o *Windows Subsystem for Linux* (WSL).

Os detalhes da configuração do Laptop Rubens são descritos a seguir:

* Notebook HP Pavilion dm4
* Memória RAM: 16 Gbytes
* SSD: 1 TBytes
* Sistema Operacional utilizado no Windows Subsystem for Linux:
  + Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86\_64)
* CPU:
  + Model name: Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz
  + Architecture: x86\_64
  + Cores: 2
  + CPU op-mode(s): 32-bit, 64-bit
  + Address sizes: 36 bits physical, 48 bits virtual
  + Byte Order: Little Endian
  + CPU(s): 4
  + Vendor ID: GenuineIntel
  + CPU family: 6
  + Thread(s) per core: 2
  + Core(s) per socket: 2
  + L1d cache: 64 KiB (2 instances)
  + L1i cache: 64 KiB (2 instances)
  + L2 cache: 512 KiB (2 instances)
  + L3 cache: 4 MiB (1 instance)

# Ferramentas experimentadas

As ferramentas utilizadas para avaliações em arquitetura de computadores foram definidas previamente na especificação do projeto cujos resultados das execuções são apresentados na sequencia.

## SPEC CPU 2017 benchmark

O SPEC CPU 2017 é um pacote de benchmark que contém a próxima geração de SPECs (*Standard Performance Evaluation Corporation*), pacotes de processamento intensivo de CPU para medição e comparação de desempenho computacional, sobrecarregando o processador do sistema, memória e compilador. Esta ferramenta oferece 4 suites para benchmark considerando velocidade (*speed*) e throughput (*rate*) para números inteiros e em ponto flutuante sendo

* As suites SPECspeed 2017 Integer e SPECspeed 2017 Floating Point são utilizadas para comparação do tempo de processamento em computadores para completar tarefas simples.
* As suites SPECrate 2017 Integer e SPECrate 2017 Floating Point medem a taxa de transferencia (*throughput*) ou o trabalho por unidade de tempo.

### Instalação e configuração

O procedimento de instalação e configuração foi realizado seguindo as orientações contidas no site da ferramenta (<https://www.spec.org/cpu2017/Docs/quick-start.html>) a partir do software obtido previamente.

Para instalação da ferramenta foram realizados os seguintes principais passos:

* Descompactação do arquivo “spec-cpu-2017.zip”
* Compilação do SPCE CPU 2017:
  + cd /usr/local/spec\_cpu2017
  + source shrc
  + make

A execução da ferramenta requer um arquivo de configuração específico o qual pode ser produzido a partir de alguns modelos (templates) conforme o ambiente de experimentação. O modelo de configuração utilizado foi “Example-gcc-linux-x86.cfg” a partir do qual foi criado o arquivo de configuração denominado “rubens-try1.cfg” onde alguns parâmetros foram ajustados para os testes como diretórios de trabalho, flags de compilação e de otimização. Para cada suíte de teste, o número de cópias e de threads foi alterado a fim de se executar o benchmark em diversas configurações.

### Execução

Os benchmarks foram executados buscando explorar as quatro suítes disponíveis cujos comandos com os parâmetros são apresentados Tabela 1. Alguns programas não foram possíveis serem executados e, portanto, foram excluídos dos testes.

|  |  |
| --- | --- |
| **Suite** | **Comando para execução do SPEC CPU 2017** |
| intspeed | runcpu --config=rubens-try1 --noreportable --iterations=3 600.perlbench\_s 602.gcc\_s 605.mcf\_s 620.omnetpp\_s 623.xalancbmk\_s 625.x264\_s 631.deepsjeng\_s 641.leela\_s 648.exchange2\_s 998.specrand\_is |
| intrate | runcpu --config=rubens-try1 --reportable --iterations=3 intrate |
| fpspeed | runcpu --config=rubens-try1 --noreportable --iterations=3 603.bwaves\_s 607.cactuBSSN\_s 619.lbm\_s 621.wrf\_s 628.pop2\_s 638.imagick\_s 644.nab\_s 649.fotonik3d\_s 654.roms\_s 996.specrand\_fs |
| fprate | runcpu --config=rubens-try1 --reportable --iterations=3 fprate |

Tabela 1. Comandos SPEC CPU 2017 executados para as suites intspeed, intrate, fpspeed e fprate.

### Resultados

A Tabela 2 apresenta o resumo da experimentação do SPEC CPU 2017 no Laptop Rubens indicando os valores dos parâmetros da execução, a duração da execução e a métrica final de execução produzida pela ferramenta. Os resultados detalhados desse experimento podem ser consultados na seção [SPEC CPPU 2017 do repositório Github](https://github.com/rubenscp/RCP-MO601-Project-03/tree/main/spec_cpu_2017/results%20laptop%20Rubens).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Resultados da execução do SPEC CPU 2017** | | | | | | |
| **Suíte** | **Cópias** | **Threads** | **Nº Iterações** | **Qtde de Benchmarks** | **Tempo de execução** | **Métrica Final (base)** |
| intspeed | 4 | 4 | 3 | 9 | 17993 s - 4,99 hs | **3,42** |
| intspeed | 8 | 8 | 3 | 10 | 18438 s – 5,12 hs | **3,35** |
| intspeed | 16 | 16 | 3 | 10 | 32523 s - 9,03 hs | **1,96** |
| intrate | 4 | 4 | 3 | 10 | 38073 s - 10,57 hs | **5,32** |
| intrate | 8 | 8 | 3 | 9 | 65121 s – 18,08 hs | **5,03** |
| fpspeed | 4 | 4 | 3 | 9 | 79708 s - 22,14 hs | **3,11** |
| fprate | 4 | 4 | 3 | 13 | 58396 s - 16,22 hs | **6,25** |
| fprate | 8 | 1 | 3 | 14 | 124885 s - 34,69 hs | **5,82** |
| Duração total das execuções | | | | | 435137 s – 120,87 hs |  |

Tabela 2. Suites executadas na ferramenta SPEC CPU 2017 com seus parâmetros da execução, o tempo de execução e a métrica final da execução.

A Tabela 3 apresenta a comparação das métricas produzidas no Laptop Rubens e de outros computadores selecionados a partir da lista de resultados disponíveis no site oficial da ferramenta SPEC CPU 2017 (<https://www.spec.org/cpu2017/results/cpu2017.html>). Os computadores selecionados são aqueles que mais se aproximam das características do computador Laptop Rubens a fim de que as comparações possam ser equilibradas e justas. Destaca-se que as métricas finais de Laptop Rubens foram obtidas em execuções sem exclusiva do computador, isto é, diversas outras tarefas não relacionadas aos benchmarks eram executadas simultaneamente.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Suite** | **Threads** | **Métrica obtida do Laptop Rubens** | **Outros computadores** | **Métrica** |
| intspeed | 4 | int\_base: 3,42 | SuperWorkstation 5039C-T (X11SCA , Intel Core i3-8100) | int\_base: 7,58 |
| intspeed | 8 | Int\_base: 3,35 | SuperWorkstation 5039C-T (X11SCA , Intel Core i7-9700K) | int\_base: 10,6 |
| intspeed | 16 | int\_base: 1,96 | Não localizado computador equivalente com thread = 16 | --- |
| intrate | 4 | int\_base: 5,32 | ASUS Z170M-PLUS Motherboard (Intel Core i7-6700K) | int\_base: 23,5 |
| intrate | 8 | int\_base: 5,03 | SuperWorkstation 5039C-T (X11SCA , Intel Core i7-9700K) | int\_base: 44,8 |
| fpspeed | 4 | fp\_base: 3,11 | SuperWorkstation 5039C-T (X11SCA , Intel Core i7-9700K) | fp\_base: 32,2 |
| fprate | 4 | fp\_base: 6,25 | SuperWorkstation 5039C-T (X11SCA , Intel Core i7-9700K) | fp\_base: 42,6 |
| fprate | 8 | fp\_base: 5,82 | SuperWorkstation 5039C-T (X11SCA , Intel Core i7-9700K) | fp\_base: 42,6 |

Tabela 3. Comparação das métricas dos benchmarks executados no laptop Rubens e outros computadores.

## Simulador multi-core Sniper

Sniper é uma ferramenta de simulação de código voltada para a modelagem e análise do desempenho de sistemas multi-core explorando o comportamento do sistema para sua otimização.

Os resultados detalhados desse experimento podem ser consultados na seção [Sniper do repositório Github](https://github.com/rubenscp/RCP-MO601-Project-03/tree/main/sniper).

### Instalação e configuração

A instalação foi realizada conforme as instruções contidas no site do simulador (<https://snipersim.org/w/Getting_Started>) conforme a sequencia abaixo.

* + Download da ferramenta:

wget <http://snipersim.org/download/dfbc471f39ee1a74/packages/sniper-latest.tgz>

* + Atualização de bibliotecas e compilação:
  + sudo apt-get update
  + sudo apt-get install libbz2-dev
  + sudo apt-get install bzip2-devel
  + sudo apt-get install libboost-dev
  + sudo apt-get install boost-devel
  + sudo apt-get install libsqlite3-dev
  + sudo apt-get install sqlite-devel
  + sudo apt-get install xsltproc libxmu-dev
  + make -j 2
* Instalação programas de benchmark adicionais:
  + wget http://snipersim.org/packages/sniper-benchmarks.tbz
  + tar xjf sniper-benchmarks.tbz
  + cd benchmarks
  + export GRAPHITE\_ROOT =/Path/to/sniper
  + export BENCHMARKS\_ROOT=$(pwd)

### Execução

O Sniper oferece uma coleção de programas de teste cujo execução é realizada por intermédio de scripts previamente preparados. Assim, a execução de um programa pode ser resumida nos passos seguintes:

* Acesso à pasta do programa a ser executado:
  + cd /usr/local/sniper/test/fft#
* Execução do simulador Sniper para o programa escolhido:
  + make run
* Coleta dos resultados gerados.

Dentre os programas de benchmarks adicionais ao Sniper, foram selecionados o RADIX e CHOLESKY com alguns arquivos de entrada cujos seguintes comandos de execução foram conforme segue:

* Acesso à pasta dos programas adicionais:
  + cd /usr/local/sniper/projeto03-programs
* Execução dos programas:
  + time ../../run-sniper ./RADIX
  + time ../../run-sniper ./CHOLESKY input/tk14.O
  + time ../../run-sniper ./CHOLESKY input/d750.O

Para a obtenção dos tempos de execução na forma nativa, os seguintes comandos foram utilizados:

* time ./RADIX
* time ./CHOLESKY input/tk14.O
* time ./CHOLESKY input/d750.O
* time ./fft

### Resultados

A Tabela 4 indica os programas selecionados com os tempos de execução do simulador Sniper e de forma nativa com o cálculo do *slowdown* da simulação. Os dados foram extraídos por meio do comando Linux “time” e o tempo utilizado no cálculo foi o “user”, isto é, o tempo da CPU gasto exclusivamente na execução das instruções dos programas expresso em segundos. A Tabela 5 apresenta outras métricas produzidas pelo Sniper.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **Número de instruções simuladas no Sniper** | **Tempo de execução no simulador Sniper**  **(TSni)** | **Tempo de execução nativo “Total Time”**  **(TNat)** | **Slowdown de simulação**  **(TSni / TNat)** |
| radix | 46.7M | 126,044 | 0,037 | 3406,595 |
| cholesky tk14.O | 44.3M | 134,326 | 0,013 | 10332,769 |
| cholesky d750.O | 309,8M | 763,810 | 0,065 | 11750,923 |
| fft | 1.2M | 4,875 | 0,002 | 243,75 |

Tabela 4. Cálculo de slowdown para programas selecionados.

|  |  |  |  |
| --- | --- | --- | --- |
| **Programa** | **Número de Instruções Executadas no Sniper** | **Instruções por Ciclo (IPC)** | **Ciclos** |
| radix | 46.7M | 0,45 | 104,5M |
| cholesky tk14.O | 44,3M | 2,30 | 19,3M |
| cholesky d750.O | 309,8M | 1,81 | 171,2M |
| fft | 1,6M | 1,45 | 1,1M |

Tabela 5. Outras métricas de desempenho coletadas pelo Sniper.

## Perf profiler

Perf profiler é uma ferramenta Linux que coleta e analisa dados de desempenho de programas ou do sistema operacional.

### Instalação e configuração

O Perf é um utilitário nativo do Linux e na versão Ubuntu ele já estava disponível, cuja versão é a 5.15.90.1.g4aeb7776ebf6.

### Execução

A execução do Perf para os programas selecionados foi conforme segue:

* Acesso à pasta dos programas a serem avaliados pelo Perf:
  + cd /usr/local/perf-benchmark/
* Execução do Perf:
  + time perf stat ./RADIX
  + time perf stat ./CHOLESKY input/tk14.O
  + time perf stat ./CHOLESKY input/d750.O
  + time perf stat ./fft

### Resultados

A Tabela 6 apresenta as mesmas métricas coletadas de forma nativa pelo Perf aplicados aos programas RADIX, CHOLESKY e FFT, sendo as mesmas métricas coletadas pelo Sniper.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **Número de Instruções Executadas pelo Perf** | **Instruções por Ciclo**  **(IPC)** | **Ciclos** | **Tempo de execução do Perf**  **(time > user)** |
| radix | 48932918 ~ 49M | 0,45 | 109497100 ~ 109M | 0,048 |
| cholesky tk14.O | 47234927 ~ 47M | 1,46 | 32377847 ~ 32,4M | 0,029 |
| cholesky d750.O | 319675663 ~ 319,7M | 1,80 | 177815525 ~ 177,8M | 0,063 |
| fft | 2803179 ~ 2,8M | 0,71 | 3938641 ~ 3,9M | 0,003 |

Tabela 6. Métricas coletadas pelo Perf.

A comparação entre as métricas produzidas pelo Sniper e Perf demonstrou que o comportamento em ambas ferramentas está coerente sendo que:

* para a métrica “Número de Instruções Executadas”, Sniper apresentou valores menores do que Perf em todos os programas;
* para a métrica “Instruções por Ciclo (IPC)”, Sniper apresentou valores maiores do que Perf em todos os programas;
* para a métrica “Ciclos”, Sniper apresentou valores menores do que Perf em todos os programas;
* para a métrica “Tempo de Execução” na ferramenta, Sniper apresentou valores muito menores do que Perf em todos os programas.

Assim essas diferenças nas métricas entre as duas ferramentas ocorre, pois o Sniper realiza a simulação de uma arquitetura sobre determinadas configurações e o Perf colhe os dados de uma execução do programa.

## PARSEC Benchmark Suite 3.0

O PARSEC (*Princeton Application Repository for Shared-Memory Computers*) é um conjunto de benchmark composto por programas *multithread* com o propósito de possibilitar estudos de desempenho em computadores com múltiplos processadores.

### Instalação e configuração

A instalação do Parsec 3.0 foi realizada conforme as instruções na documentação da ferramenta disponíveis no site:

* Visão geral: https://parsec.cs.princeton.edu/overview.htm
* Download da ferramenta: <https://parsec.cs.princeton.edu/download/3.0/>

Alguns ajustes foram necessários para a compilação como a adição do flag –std=c++11 no arquivo gcc.bldconf localizado na pasta config.

Antes de iniciar a compilação, execute os passos a seguir:

* cd /usr/local/parsec-3.0
* source env.sh

A compilação dos programas (apps) que acompanham a ferramenta foi realizada e o resultado é na Tabela 7 indicando o sucesso na compilação ou o erro ocorrido.

|  |  |  |
| --- | --- | --- |
| **Pacote** | **Comando para compilação** | **Resultado da Compilação** |
| blackscholes | parsecmgmt -a build -p blackscholes | Sucesso |
| bodytrack | parsecmgmt -a build -p bodytrack | Sucesso |
| fluidanimate | parsecmgmt -a build -p fluidanimate | Sucesso |
| freqmine | parsecmgmt -a build -p freqmine | Sucesso |
| vips | parsecmgmt -a build -p vips | Sucesso |
| facesim | parsecmgmt -a build -p facesim | make[2]: \*\*\* [/usr/local/parsec-3.0/pkgs/apps/facesim/obj/amd64-linux.gcc/Public\_Library/Makefile.common:407: obj/Collisions\_And\_Interactions/COLLISION\_BODY\_LIST\_3D.o] Error 1  make[2]: Leaving directory '/usr/local/parsec-3.0/pkgs/apps/facesim/obj/amd64-linux.gcc/Public\_Library'  make[1]: \*\*\* No rule to make target '/usr/local/parsec-3.0/pkgs/apps/facesim/obj/amd64-linux.gcc/lib/libPhysBAM.a', needed by 'facesim'. Stop.  make[1]: Leaving directory '/usr/local/parsec-3.0/pkgs/apps/facesim/obj/amd64-linux.gcc/Benchmarks/facesim'  make: \*\*\* [Makefile:16: all] Error 2  [PARSEC] Error: 'env version=pthreads PHYSBAM=/usr/local/parsec-3.0/pkgs/apps/facesim/obj/amd64-linux.gcc CXXFLAGS=-O3 -g -funroll-loops -fprefetch-loop-arrays -fpermissive -fno-exceptions -std=c++11 -static-libgcc -Wl,--hash-style=both,--as-needed -DPARSEC\_VERSION=3.0-beta-20150206 -fexceptions /usr/bin/make' failed. |
| ferret | parsecmgmt -a build -p ferret | make: \*\*\* [Makefile:108: /usr/local/parsec-3.0/pkgs/apps/ferret/obj/amd64-linux.gcc/parsec/obj/LSH\_query.o] Error 1  [PARSEC] Error: 'env version=pthreads CFLAGS=-I/usr/local/parsec-3.0/pkgs/libs/gsl/inst/amd64-linux.gcc/include -I/usr/local/parsec-3.0/pkgs/libs/libjpeg/inst/amd64-linux.gcc/include -O3 -g -funroll-loops -fprefetch-loop-arrays -static-libgcc -Wl,--hash-style=both,--as-needed -DPARSEC\_VERSION=3.0-beta-20150206 LDFLAGS=-L/usr/local/parsec-3.0/pkgs/libs/gsl/inst/amd64-linux.gcc/lib -L/usr/local/parsec-3.0/pkgs/libs/libjpeg/inst/amd64-linux.gcc/lib -L/usr/lib64 -L/usr/lib /usr/bin/make' failed. |
| raytrace | parsecmgmt -a build -p raytrace | No package 'xext' found  Consider adjusting the PKG\_CONFIG\_PATH environment variable if you  installed software in a non-standard prefix.  Alternatively, you may set the environment variables XLIBGL\_CFLAGS  and XLIBGL\_LIBS to avoid the need to call pkg-config.  See the pkg-config man page for more details.  [PARSEC] Error: 'env ./configure --with-driver=xlib --enable-glut --enable-static --disable-shared --prefix=/usr/local/parsec-3.0/pkgs/libs/mesa/inst/amd64-linux.gcc' failed. |
| swaptions | parsecmgmt -a build -p swaptions | | ^~~~~~  make[1]: \*\*\* [../../build/Makefile.tbbmalloc:70: proxy.o] Error 1  make[1]: Leaving directory '/usr/local/parsec-3.0/pkgs/libs/tbblib/obj/amd64-linux.gcc/build/linux\_intel64\_gcc\_cc11.3.0\_libc2.35\_kernel5.15.90.1\_debug'  make: \*\*\* [Makefile:49: tbbmalloc] Error 2  [PARSEC] Error: 'env compiler=gcc PATH=/usr/bin:/usr/local/parsec-3.0/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/usr/local/parsec-3.0/bin CXXFLAGS=-O3 -g -funroll-loops -fprefetch-loop-arrays -fpermissive -fno-exceptions -static-libgcc -Wl,--hash-style=both,--as-needed -DPARSEC\_VERSION=3.0-beta-20150206 -fexceptions /usr/bin/make' failed. |

Tabela 7. Resultado da compilação dos programas no PARSEC 3.0.

### Execução

O PARSEC possibilita definir a região de interesse (ROI – *Region Of Interest*) na execução dos benchmarks baseado em seis tipos de entrada possíveis: *test, simdev, simsmall, simmedium, simlarge* e *native*. Além disso, existem outros parâmetros como “-n” que indica o número mínimo de threads na execução. O valor default de “n” é 1.

A execução dos programas no Parsec foi realizada com o parâmetro “native” e variando o número de threads (N) cujos comandos de execução estão apresentados na Tabela 13 do apêndice desse relatório.

Os resultados detalhados desse experimento podem ser consultados na seção [Parsec do repositório Github](https://github.com/rubenscp/RCP-MO601-Project-03/tree/main/parsec/result).

### Resultados

Para avaliação dos resultados da execução, foi utilizada a métrica do tempo de execução das instruções do programa (*user*) com entrada nativa (*native*) e explorando diversos números de threads (1, 2, 4, 8 e 16).

A Tabela 8 apresenta os tempos de execução dos programas executados pelo Parsec com entrada “native” combinado com variado número de threads.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Programa** | **thread = 1** | **thread = 2** | **thread = 4** | **thread = 8** | **thread = 16** |
| blackscholes | 119,691 s | 82,85 s | 72,226 s | 75,989 s | 90,066 s |
| vips | 133,075 s | 88,737 s | 76,509 s | 79,51 s | 82,772 s |
| bodytrack | 151,276 s | 98,623 s | 86,545 s | 86,493 s | 90,33 s |
| fluidanimate | 405,59 s | 320,64 s | 280,12 s | 286,658 s | 293,435 s |
| freqmine | 609,146 s | 395,635 s | 324,965 s | 313,262 s | 351,112 s |

Tabela 8. Execução do Parsec nos programas de benchmark com o parâmetro de entrada “-i native” combinado com os diversos valores de número de Threads “-n “ apresentando os tempos de execução.

Figura 1. Gráfico do Tempo de execução (real) das instruções dos programas em função do número de threads.

## Rodinia benchmark

O Rodinia Benchmark é uma ferramenta destinada a infraestrutura de computação heterogênea com implementações com OpenMP, OpenCL e CUDA.

### Instalação e configuração

A instalação do Rodinia ocorreu conforme as instruções na documentação da ferramenta disponíveis no site e no documento “README”:

* Visão geral: <https://rodinia.cs.virginia.edu/doku.php>
* Download da ferramenta: <http://lava.cs.virginia.edu/Rodinia/download.htm>

A instalação do CUDA foi realizada seguinte os passos a seguir:

* Site: https://linuxhint.com/install-cuda-on-ubuntu-22-04-lts/
  + sudo apt update
  + sudo apt install build-essential
  + gcc --version
  + g++ --version
  + sudo apt install nvidia-cuda-toolkit nvidia-cuda-toolkit-gcc
  + nvcc –version – a checagem da instalação é feita por meio da mensagem abaixo.
    - “CUDA version 11.5 is installed on our Ubuntu machine.”
* Ajuste no make.config indicando o path para os binários do CUDA:
  + vi common/make.config
    - alterar CUDA\_DIR = /usr

A compilação do Rodinia foi realizada por meio dos comandos que seguem:

* make clean
* make CUDA
* make OMP
* make OPENCL

### Execução

A execução do Rodínia foi realizada acessando cada pasta de benchmark e executando o script destinado a esse fim. O resultado da execução de um benchmark é um arquivo denominado result.txt:

Acessar a pasta do Rodinia:

* cd /usr/local/rodinia\_3.1

Execução do Rodínia nos programas que compilaram com sucesso em cada especificação:

* Acesso à pasta da especificação opemmp com a execução do programa e coleta de dados:
  + cd openmp
  + cd bfs depois time ../run
  + cd ../cfd depois time ../run
  + cd ../heartwall depois time ../run
  + cd ../hotspot depois time ../run
  + cd ../kmeans depois time ../run
  + cd ../lavaMD depois time ../run
  + cd ../leukocyte depois time ../run
  + cd ../nn depois time ../run
  + cd ../particle\_filter depois time ../run
  + cd ../pathfinder depois time ../run
  + cd ../srad\_v1 depois time ../run
  + cd ../srad\_v2 depois time ../run
* Acesso à pasta da especificação cuda com a execução do programa e coleta de dados:
  + cd cuda
  + cd bfs depois time ../run
  + cd ../heartwall depois time ../run
  + cd ../hotspot depois time ../run

### Resultados

A Tabela 9 apresenta os tempos de execução das instruções dos programas possíveis nas implementações OpenMp e Cuda.

|  |  |  |
| --- | --- | --- |
| **Programa** | **Tempo de execução do programa (time > user)** | |
| **Opemmp** | **Cuda** |
| bfs | 0m2.164s | 0m1.718s |
| heartwall | 0m35.370s | 0m0.154s |
| hotspot | 0m1.630s | 0m0.293s |
| cfd | 16m39.135s | --- |
| kmeans | 0m0.001s | --- |
| lavaMD | 0m11.815s | --- |
| leukocyte | 0m41.787s | --- |
| nn | 0m4.560s | --- |
| particle\_filter | 0m1.368s | --- |
| pathfinder | 0m2.436s | --- |
| srad\_v1 | 0m4.153s | --- |
| srad\_v2 | 0m0.588s | --- |

Tabela 9. Execução do Rodínia em programas das implementações OpenMp e Cuda com os respectivos tempos de execução.

## Intel Pin

Intel Pin é uma ferramenta de instrumentação dinâmica para as arquiteturas do conjunto de instruções IA-32, x86-64 e MIC permitindo a análise de programas.

### Instalação e configuração

A instalação do Intel Pin ocorreu conforme as orientações contidas do documento “README”:

* Site oficial: <https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html>
* Download:
  + wget <https://software.intel.com/sites/landingpage/pintool/downloads/pin-3.27-98718-gbeaa5d51e-gcc-linux.tar.gz>

### Execução

Para a execução, foram selecionadas algumas ferramentas do PinTools conforme segue:

* Acesso à pasta raiz do Intel Pin:
  + cd /usr/local/pin-3.27-98718-gbeaa5d51e-gcc-linux/
* Ferramenta inscount0.so localizado na pasta ManualExamples:
  + ../../../pin -t obj-intel64/inscount0.so -- ../../../projeto03-programs/radix/RADIX
  + ../../../pin -t obj-intel64/inscount0.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/tk14.O
  + ../../../pin -t obj-intel64/inscount0.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/d750.O
  + ../../../pin -t obj-intel64/inscount0.so -- ../../../projeto03-programs/fft/fft
* Ferramenta opcodemix localizado na pasta SimpleExamples:
  + ../../../pin -t obj-intel64/opcodemix.so -- ../../../projeto03-programs/radix/RADIX
  + ../../../pin -t obj-intel64/opcodemix.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/tk14.O
  + ../../../pin -t obj-intel64/opcodemix.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/d750.O
  + ../../../pin -t obj-intel64/opcodemix.so -- ../../../projeto03-programs/fft/fft
* Ferramenta proccount.so localizado na pasta ManualExamples:
  + ../../../pin -t obj-intel64/proccount.so -- ../../../projeto03-programs/radix/RADIX
  + ../../../pin -t obj-intel64/proccount.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/tk14.O
  + ../../../pin -t obj-intel64/proccount.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/d750.O
  + ../../../pin -t obj-intel64/proccount.so -- ../../../projeto03-programs/fft/fft
* Ferramenta pinatrace.so localizado na pasta ManualExamples:
  + ../../../pin -t obj-intel64/pinatrace.so -- ../../../projeto03-programs/radix/RADIX
  + ../../../pin -t obj-intel64/pinatrace.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/tk14.O
  + ../../../pin -t obj-intel64/pinatrace.so -- ../../../projeto03-programs/cholesky/CHOLESKY ../../../projeto03-programs/cholesky/input/d750.O
  + ../../../pin -t obj-intel64/pinatrace.so -- ../../../projeto03-programs/fft/fft

### Resultados

A Tabela 10 apresenta os resultados da execução das ferramentas PinTools aplicadas aos programas selecionados nas ferramentas anteriores. Cada ferramenta apresenta um resultado específico de acordo com sua funcionalidade.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **inscount0** | **opcodemix** | **proccount** | **pinatrace** |
| Realiza a contagem de instruções executadas pelo programa | Produz estatística do número de vezes instruções foram executadas | Produz estatística do número de chamadas às funções ou procedimentos do programa | Cria arquivo de *trace* do programa |
| radix | 46644602 ~ 46,6M | 396 instruções | 83 procedimentos | 4810142 linhas de trace ~ 4,8M |
| cholesky tk14.O | 44295564 ~ 44M | 465 instruções | 185 procedimentos | 13715252 linhas de trace ~ 13,7M |
| cholesky d750.O | 309828473 ~ 309M | 465 instruções | 180 procedimentos | 97326228 linhas de trace ~ 97M |
| fft | 1667517 ~ 166 mil | 441 instruções | 184 procedimentos | 834499 linhas de trace ~ 834mil |

Tabela 10. Comandos para execução do PinTools nos programas selecionados com os respectivos resultados.

## Dinero cache simulator

A ferramenta Dinero é um simulador de cache de 4ª geração de simuladores.

Os programas utilizados nessa ferramenta foram o RADIX e o fft. Vários parâmetros foram avaliados considerando valores distintos para cache L1 (instrução e data), combinados com cache L2 e L3 (unificadas).

### Instalação e configuração

A instalação do DineroIV foi realizada observando as instruções contidas no site da ferramenta:

* Site oficial: <https://pages.cs.wisc.edu/~markhill/DineroIV/>
* Site complementar:
  + <http://thedarklair.free.fr/prague/2006/XE36APS%20-%20Architecture%20Of%20Computer%20Systems/Seminars/10/cache_en.htm>

A instalação do DineroIV foi realizada segundo os passos abaixo:

* + Download:
    - git clone <https://github.com/zjutoe/DineroIV.git>
  + Build
    - Acessar a pasta d4-7
    - ./configure
    - make

### Execução

Para a experimentação no dineroIV, foi utilizado o programa RADIX no formato “din” requerido pela ferramenta, conforme as etapas que seguem:

* Geração do arquivo de trace do programa RADIX por meio do PinTool pinatrace:
  + ../../../pin -t obj-intel64/pinatrace.so -- ../../../projeto03-programs/radix/RADIX
* Conversão do arquivo de trace gerado na etapa anterior para o formato din por meio de um programa conversor python, o qual realiza o reposicionamento dos campos (tipo de acessos, endereço e tamanho) para posterior execução pelo simulador dineroIV:
  + python3 gera\_arquivo\_din.py

O código fonte do programa gera\_arquivo\_din.py é apresentado a seguir.

|  |
| --- |
| # Conversor from trace file (pinatrace.so) to din format required to DineroIV benchmark tool.  input\_filename = 'radix.trace.out'  output\_filename = 'radix.din'  output\_file = open(output\_filename, 'w', newline='\n', encoding='utf-8')  with open(input\_filename, 'r') as file:  text = file.readlines()  count = 0  for input\_line in text:  try:  tokens = input\_line.split()  output\_line = tokens[1] + " " + tokens[0].replace(':', '') \  + " " + tokens[2] + "\n"  output\_file.writelines(output\_line)  count += 1  except Exception:  pass  file.close() |

* Execução do dineroIV utilizando como entrada o arquivo no formato “din” gerado na etapa anterior e em diversas configurações de cache L1, L2 e L3.

./dineroIV   
-l1-isize [P] -l1-dsize [P] -l1-ibsize [B] -l1-dbsize [B] -l1-iassoc [N] -l1-dassoc[N]   
-l2-usize [P] -l2-ubsize [B] -l2-uassoc [N]   
-l3-usize [P] -l3-ubsize [B] -l3-uassoc [N] -informat p   
< [program.din] > results/dinero-result-[programa.din]-EEE.txt

Nessa ferramenta foram testadas diferentes configurações de cache l1, l2 e l3 a fim de se obter a melhor configuração. As configurações com os diferentes tamanhos das caches estão descritas na tabela abaixo e para a escolha foi utilizada a métrica *demand miss rate*

### Resultados

Para uma melhor avaliação dos resultados experimentados, a configuração do Laptop Rubens possui a seguinte configuração de cache:

* + L1d cache: 64 KB
  + L1i cache: 64 KB
  + L2 cache: 512 KB
  + L3 cache: 4 MB

Os resultados obtidos das diversas execuções foram

A Tabela 11 apresenta os comandos utilizados na execução dos programas RADIX e fft com os diversos parâmetros de execução relacionados às caches L1, L2 e L3.

| **#** | **L1 cache** | | | | | | **L2 cache** | | | **L3 cache** | | | **Demand Miss Rate** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **isize** | **dsize** | **ibsize** | **dbsize** | **iassoc** | **dassoc** | **usize** | **ubsize** | **uassoc** | **usize** | **ubsize** | **uassoc** | **L1 instr** | **L1 dados** | **L2**  **unif** | **L3**  **Unif** |
| 1 | 1k | 1k | 2 | 2 |  |  |  |  |  |  |  |  | 0,6667 | 0,3749 | x | x |
| 2 | 1k | 1k | 4 | 4 |  |  |  |  |  |  |  |  | 0,5 | 0,463 | x | x |
| 3 | 1k | 1k | 8 | 8 |  |  |  |  |  |  |  |  | 0,3333 | 0,5548 | x | x |
| 4 | 1k | 1k | 16 | 16 |  |  |  |  |  |  |  |  | 0,2 | 0,5654 | x | x |
| 5 | 1k | 1k | 32 | 32 |  |  |  |  |  |  |  |  | 0,1111 | 0,6258 | x | x |
| 6 | 1k | 1k | 32 | 2 |  |  |  |  |  |  |  |  | 0,1111 | 0,3749 | x | x |
| 7 | 2k | 2k | 32 | 2 |  |  |  |  |  |  |  |  | 0,1111 | 0,1926 | x | x |
| 8 | 4k | 4k | 32 | 2 |  |  |  |  |  |  |  |  | 0,1111 | 0,1822 | x | x |
| 9 | 8k | 8k | 32 | 2 |  |  |  |  |  |  |  |  | 0,1111 | 0,1367 | x | x |
| 10 | 16k | 16k | 32 | 2 |  |  |  |  |  |  |  |  | 0,1111 | 0,1367 | x | x |
| 11 | 32k | 32k | 32 | 2 |  |  |  |  |  |  |  |  | 0,1111 | 0,1342 | x | x |
| 12 | 32k | 32k | 32 | 2 | 2 | 2 |  |  |  |  |  |  | 0,1111 | 0,0829 | x | x |
| 13 | 32k | 32k | 32 | 2 | 4 | 4 |  |  |  |  |  |  | 0,1111 | 0,0725 | x | x |
| 14 | 32k | 32k | 32 | 2 | 8 | 8 |  |  |  |  |  |  | 0,1111 | 0,0555 | x | x |
| 15 | 32k | 32k | 32 | 2 | 16 | 16 |  |  |  |  |  |  | 0,1111 | 0,0282 | x | x |
| 16 | 32k | 32k | 32 | 2 | 32 | 32 |  |  |  |  |  |  | 0,1111 | 0,0325 | x | x |
| 17 | 32k | 32k | 32 | 2 | 32 | 32 | 1k | 2 | 32 |  |  |  | 0,1111 | 0,0325 | 0,9999 | x |
| 18 | 32k | 32k | 32 | 2 | 32 | 32 | 2k | 2 | 32 |  |  |  | 0,1111 | 0,0325 | 0,9999 | x |
| 18 | 32k | 32k | 32 | 2 | 32 | 32 | 4k | 2 | 32 |  |  |  | 0,1111 | 0,0325 | 0,9998 | x |
| 20 | 32k | 32k | 32 | 2 | 32 | 32 | 8k | 2 | 32 |  |  |  | 0,1111 | 0,0325 | 0,9996 | x |
| 21 | 32k | 32k | 32 | 2 | 32 | 32 | 16k | 2 | 32 |  |  |  | 0,1111 | 0,0325 | 0,9994 | x |
| 22 | 32k | 32k | 32 | 2 | 32 | 32 | 32k | 2 | 2 |  |  |  | 0,1111 | 0,0325 | 0,9997 | x |
| 23 | 32k | 32k | 32 | 2 | 32 | 32 | 32k | 2 | 4 |  |  |  | 0,1111 | 0,0325 | 0,9995 | x |
| 24 | 32k | 32k | 32 | 2 | 32 | 32 | 32k | 2 | 8 |  |  |  | 0,1111 | 0,0325 | 0,9992 | x |
| 25 | 32k | 32k | 32 | 2 | 32 | 32 | 32k | 2 | 16 |  |  |  | 0,1111 | 0,0325 | 0,999 | x |
| 26 | 32k | 32k | 32 | 2 | 32 | 32 | 32k | 2 | 32 |  |  |  | 0,1111 | 0,0325 | 0,9991 | x |
| 27 | 32k | 32k | 32 | 2 | 32 | 32 | 64k | 2 | 16 |  |  |  | 0,1111 | 0,0325 | 0,9977 | x |
| 28 | 32k | 32k | 32 | 2 | 32 | 32 | 128k | 2 | 16 |  |  |  | 0,1111 | 0,0325 | 0,9969 | x |
| 29 | 32k | 32k | 32 | 2 | 32 | 32 | 256k | 2 | 16 |  |  |  | 0,1111 | 0,0325 | 0,9958 | x |
| 30 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 2 | 16 |  |  |  | 0,1111 | 0,0325 | 0,9947 | x |
| 31 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 2 | 16 | 1m | 2 |  | 0,1111 | 0,0325 | 0,9947 | 0,999 |
| 32 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 2 | 16 | 2m | 2 |  | 0,1111 | 0,0325 | 0,9947 | 0,998 |
| 33 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 2 | 16 | 4m | 2 |  | 0,1111 | 0,0325 | 0,9947 | 0,9979 |
| 34 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 2 | 16 | 4m | 2 | 16 | 0,1111 | 0,0325 | 0,9947 | 0,9978 |
| 35 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 2 | 32 | 4m | 2 | 32 | 0,1111 | 0,0325 | 0,9927 | 0,9999 |
| 36 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 32 | 16 | 4m | 32 | 16 | 0,1111 | 0,0325 | 0,9037 | 0,9776 |
| 37 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 256 | 16 | 4m | 256 | 16 | 0,1111 | 0,0325 | 0,1222 | 0,8454 |
| 38 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 512 | 128 | 4m | 512 | 128 | 0,1111 | 0,0325 | 0,0559 | 0,9989 |
| 39 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 1k | 128 | 4m | 1k | 128 | 0,1111 | 0,0325 | 0,0279 | 0,9992 |
| 40 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 4k | 128 | 4m | 4k | 128 | 0,1111 | 0,0325 | 0,007 | 0,999 |
| 41 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 4k | 128 | 4m | 8k | 128 | 0,1111 | 0,0325 | 0,007 | 0,4998 |
| 42 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 4k | 128 | 4m | 16k | 128 | 0,1111 | 0,0325 | 0,007 | 0,2501 |
| 43 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 4k | 128 | 4m | 32k | 128 | 0,1111 | 0,0325 | 0,007 | 0,1253 |
| 44 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 8k | 32 | 4m | 32k | 32 | 0,1111 | 0,0325 | 0,0035 | 0,2461 |
| 45 | 32k | 32k | 32 | 2 | 32 | 32 | 512k | 8k | 32 | 4m | 32k | 128 | 0,1111 | 0,0325 | 0,0035 | 0,2461 |

Tabela 11. Estudo de configurações de cache por meio da ferramenta Dinero Cache Simulator aplicado ao programa RADIX.

Os resultados detalhados desse experimento podem ser consultados na seção [Dinero do repositório Github](https://github.com/rubenscp/RCP-MO601-Project-03/tree/main/dinero/results).

**Decisão sobre a melhor configuração de cache entre as testadas.**

* **A melhor configuração de cache seria aquela com menor índice de miss?**

O tamanho de uma memória cache deve ser: Suficientemente pequeno para que o custo total médio por bit seja próximo do custo por bit da memória principal; Suficientemente grande para que o tempo médio de acesso à memória seja próximo ao tempo de acesso da memória cache. (http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/Org\_Arq\_I\_2008\_1/Org\_Arq\_Comp\_2008\_Aula\_05.pdf)

# Considerações sobre o aprendizado nesse projeto

Esse projeto propôs um desafio diferente dos projetos anteriores devido ao uso de ferramentas de terceiros as quais deveriam ser instaladas, configuradas, compiladas e por fim, executadas no ambiente operacional escolhido. Algumas dessas ferramentas foram disponibilizadas para determinadas versões de compiladores e bibliotecas. Devido a essa diversidade de versões, surgiram vários desafios que precisam ser transpostos para se iniciar a tarefa de coleta dados e avaliar desempenho de benchmarks e programas. No meu caso, estimo que me dediquei a mais de 50% do tempo do projeto em tarefas de configuração e compilação das ferramentas, algo que poderia ser reduzido caso existisse documentação mais amigável e direcionada a determinado sistema operacional em uma certa versão. Apesar desse tempo, muito aprendizado foi obtido ao se conseguir ter todas as sete ferramentas aptas para o uso nas avaliações.

Outro aspecto de aprendizagem é o conhecimento que se consolida com experimentos reais de simulação demonstrando a importância na área de arquitetura de computadores para a inovação. Essas ferramentas de simulação auxiliam novos projetos de processadores e de seus componentes na etapa de validação.

Por fim, creio que ao final desse projeto consegui ter uma visão mesmo que superficial dessas ferramentas com possibilidade de aprofundamento em uma ou outra ferramenta caso necessário.

# Apêndice

## PARSEC Benchmark Suite 3.0 \*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Núm. da Execução** | **Pacote** | **Entrada** | **N** | **Comando de execução do pacote de Benchmark** |
| 101 | blackscholes | native | 1 | parsecmgmt -a run -p blackscholes -i native -n 1 > results/parsec-result-101-blackscholes-native-n1.txt |
| 102 | blackscholes | native | 2 | parsecmgmt -a run -p blackscholes -i native -n 2 > results/parsec-result-102-blackscholes-native-n2.txt |
| 103 | blackscholes | native | 4 | parsecmgmt -a run -p blackscholes -i native -n 4 > results/parsec-result-103-blackscholes-native-n4.txt |
| 104 | blackscholes | native | 8 | parsecmgmt -a run -p blackscholes -i native -n 8 > results/parsec-result-104-blackscholes-native-n8.txt |
| 105 | vips | native | 1 | parsecmgmt -a run -p vips -i native -n 1 > results/parsec-result-105-vips-native-n1.txt |
| 106 | vips | native | 2 | parsecmgmt -a run -p vips -i native -n 2 > results/parsec-result-106-vips-native-n2.txt |
| 107 | vips | native | 4 | parsecmgmt -a run -p vips -i native -n 4 > results/parsec-result-107-vips-native-n4.txt |
| 108 | vips | native | 8 | parsecmgmt -a run -p vips -i native -n 8 > results/parsec-result-108-vips-native-n8.txt |
| 109 | bodytrack | native | 1 | parsecmgmt -a run -p bodytrack -i native -n 1 > results/parsec-result-109-bodytrack-native-n1.txt |
| 110 | bodytrack | native | 2 | parsecmgmt -a run -p bodytrack -i native -n 2 > results/parsec-result-110-bodytrack-native-n2.txt |
| 111 | bodytrack | native | 4 | parsecmgmt -a run -p bodytrack -i native -n 4 > results/parsec-result-111-bodytrack-native-n4.txt |
| 112 | bodytrack | native | 8 | parsecmgmt -a run -p bodytrack -i native -n 8 > results/parsec-result-112-bodytrack-native-n8.txt |
| 113 | fluidanimate | native | 1 | parsecmgmt -a run -p fluidanimate -i native -n 1 > results/parsec-result-113-fluidanimate-native-n1.txt |
| 114 | fluidanimate | native | 2 | parsecmgmt -a run -p fluidanimate -i native -n 2 > results/parsec-result-114-fluidanimate-native-n2.txt |
| 115 | fluidanimate | native | 4 | parsecmgmt -a run -p fluidanimate -i native -n 4 > results/parsec-result-115-fluidanimate-native-n4.txt |
| 116 | fluidanimate | native | 8 | parsecmgmt -a run -p fluidanimate -i native -n 8 > results/parsec-result-116-fluidanimate-native-n8.txt |
| 117 | freqmine | native | 1 | parsecmgmt -a run -p freqmine -i native -n 1 > results/parsec-result-117-freqmine-native-n1.txt |
| 118 | freqmine | native | 2 | parsecmgmt -a run -p freqmine -i native -n 2 > results/parsec-result-118-freqmine-native-n2.txt |
| 119 | freqmine | native | 4 | parsecmgmt -a run -p freqmine -i native -n 4 > results/parsec-result-119-freqmine-native-n4.txt |
| 120 | freqmine | native | 8 | parsecmgmt -a run -p freqmine -i native -n 8 > results/parsec-result-120-freqmine-native-n8.txt |
| 121 | splash2 | native | 1 | parsecmgmt -a run -p splash2 -i native -n 1 > results/parsec-result-121-splash2-native-n1.txt |
| 122 | splash2 | native | 2 | parsecmgmt -a run -p splash2 -i native -n 2 > results/parsec-result-122-splash2-native-n2.txt |
| 123 | splash2 | native | 4 | parsecmgmt -a run -p splash2 -i native -n 4 > results/parsec-result-123-splash2-native-n4.txt |
| 124 | splash2 | native | 8 | parsecmgmt -a run -p splash2 -i native -n 8 > results/parsec-result-124-splash2-native-n8.txt |
| 125 | splash2x | native | 1 | parsecmgmt -a run -p splash2x -i native -n 1 > results/parsec-result-125-splash2x-native-n1.txt |
| 126 | splash2x | native | 2 | parsecmgmt -a run -p splash2x -i native -n 2 > results/parsec-result-126-splash2x-native-n2.txt |
| 127 | splash2x | native | 4 | parsecmgmt -a run -p splash2x -i native -n 4 > results/parsec-result-127-splash2x-native-n4.txt |
| 128 | splash2x | native | 8 | parsecmgmt -a run -p splash2x -i native -n 8 > results/parsec-result-128-splash2x-native-n8.txt |
| 129 | blackscholes | native | 16 | parsecmgmt -a run -p blackscholes -i native -n 16 > results/parsec-result-129-blackscholes-native-n16.txt |
| 130 | vips | native | 16 | parsecmgmt -a run -p vips -i native -n 16 > results/parsec-result-130-vips-native-n16.txt |
| 131 | bodytrack | native | 16 | parsecmgmt -a run -p bodytrack -i native -n 16 > results/parsec-result-131-bodytrack-native-n16.txt |
| 132 | fluidanimate | native | 16 | parsecmgmt -a run -p fluidanimate -i native -n 16 > results/parsec-result-132-fluidanimate-native-n16.txt |
| 133 | freqmine | native | 16 | parsecmgmt -a run -p freqmine -i native -n 16 > results/parsec-result-133-freqmine-native-n16.txt |
| 134 | splash2 | native | 16 | parsecmgmt -a run -p splash2 -i native -n 16 > results/parsec-result-134-splash2-native-n16.txt |
| 135 | splash2x | native | 16 | parsecmgmt -a run -p splash2x -i native -n 16 > results/parsec-result-135-splash2x-native-n16.txt |

Tabela 13. Execução do PARSEC nos vários programas com variações na entrada e número de threads.