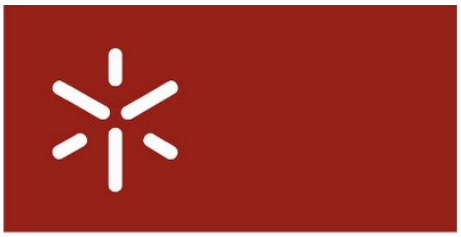
Universidade do Minho

Engenharia de Sistemas de Computação

Mestrado Integrado em Engenharia Informática



Universidade do Minho

Bernardo Silva - a77230

Francisco Lira - a73909

Portfólio

Trabalhos Práticos

Conteúdo

Capítulo 1

Introdução

Para a disciplina de Engenharia de Sistemas de Computação foram propostos 4 trabalhos práticos os quais permitem a exploração várias ferramentas de análise de performance de programas, entre elas NAS Parallel Benchmarks, DTrace e Perf.

Estas permitem ao utilizador espreitar o funcionamento interno dos programas e localizar hot spots e obter informações acerca do seu funcionamento.

De seguida iremos fazer uma análise dessas ferramentas e dos resultados obtidos com as mesmas.

Capítulo 2

TP1- NAS Parallel Benchmarks

O objetivo deste trabalho prático passa pela utilização, observação e experimentação com as NAS Parallel Benchmarks com recurso a ferramentas de *profiling* disponibilizadas pela maioria dos sistemas UNIX, utilizando máquinas do cluster Search, de forma a explorar estes ambientes de *benchmarking*.

2.1 Ambiente de execução em clustering

2.1.1 Escolha de máquinas

Para a realização da primeira fase deste trabalho prático definimos como máquinas a explorar as máquinas r641, r652 e r662 do cluster Search, as quais possuem 16, 20 e 24 cores respetivamente.

2.1.2 Algoritmos disponíveis

Estes testes exploravam as NAS Parallel Benchmarks Multizone, ou seja, a versão NPB 3.3.1-MZ das mesmas.

Consequentemente, os algoritmos disponíveis na mesma foram os seguintes:

- BT-MZ
- SP-MZ
- LU-MZ

Cada um destes algoritmos possúi, por sua vez, uma versão serial (sequencial), uma versão OpenMP e uma versão híbrida MPI+OpenMP.

As versões OpenMp requeriam que se escolhesse um número de threads interiores e exteriores. Como medida de comparação nas versões OpenMP atribuiu-se para a máquina r641 4 threads exteriores e 4 threads interiores, para a máquina r652 um teste com 5 threads exteriores e 4 threads interiores, e outro teste com 4 threads exteriores e 5 threads interiores, e finalmente para a máquina r662 um teste com 6 threads exteriores e 4 threads interiores e outro teste com 4 threads exteriores e 6 threads interiores.

Nas versões MPI+OpenMP apenas se requeria que se escolhesse um número de processos e um número de threads. Logo decidimos escolher 2 processos e metade dos threads totais suportados pela máquina. Ou seja, na máquina r641 levaria a 8 threads, na r652: 10 threads e na r662: 12 threads.

2.1.3 Tamanho de Problema

As NAS Parallel Benchmarks disponibilizam a resolução dos seus algoritmos com vários tamanhos:

Weight	Memory Requirement
S	1 MiB
W	6 MiB
A	50 MiB
B	200 MiB
C	0.8 GiB
D	12.8 GiB
E	250 GiB
F	5.0 TiB

Em termos de tamanho de problema utilizaram-se os tamanhos A, B e C como algoritmos para análise. Estes tamanhos foram utilizados para todos os algoritmos e em todas as máquinas.

2.1.4 Compilação

Para compilação utilizou-se o compilador gcc tanto para fortran como c. Como biblioteca para mpi utilizou-se OpenMPI, sempre com a flag -O3.

2.1.5 Obtenção de Resultados

Para obtenção dos resultados referentes a tempo de execução este era medido pelo benchmark em si.

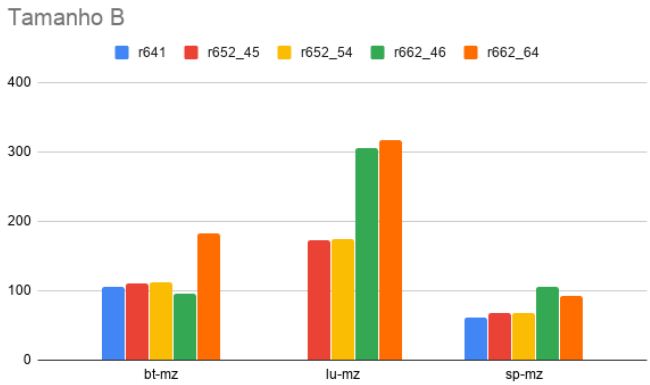
O resto dos resultados foram obtidos utilizando vmstat após 1 segundo da execução do programa utilizando o seguinte one-liner:

```
(sleep 1 && vmstat -s > vmstat.txt) & program.x
```

Um one liner similar, mas bastante mais complexo foi utilizado na segunda fase para obtenção de resultados.

2.1.6 Resultados Obtidos (Tempo de execução)

Começamos por realizar, para as diferentes classes, os diferentes benchmarks na versão com MPI, em seguida OMP e por fim a versão hibrida.



Tempo de Execução(sec) @ MPI, B

Neste primeiro gráfico podemos já destacar a diferença nos resultados do benchmark BT-MZ quando na máquina 662 temos a alteração dos threads interiores e exteriores de 4 para 5 e vice versa. Esta alteração levou a um impacto significativo no tempo de execução, aproximadamente 2 vezes superior. Já na máquina 652 esta diferença foi quase insignificante. Neste gráfico podemos ainda reparar que a máquina 662 tem tempos de execução sempre superiores o que nos leva a concluir que a divisão do trabalho por mais threads é mais demoroso do que a execução propriamente dita do algoritmo.

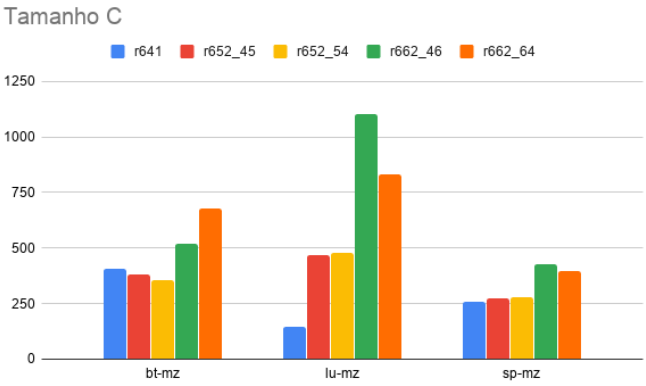
Também no tempo de execução da versão de OMP se nota a maior diferença nos resultados entre as versões de diferentes configurações das threads, já entre as diferentes máquinas a diferença é de no máximo 6% nos testes para o tamanho A que são os que apresentam maior diferença entre resultados pela menor dimensão.

Para a versão híbrida, os resultados são aqueles que esperavamos encontrar, uma vez que é um equilíbrio entre os resultados obtidos anteriormente. Mas aqui a diferença percentual entre os diferentes tamanhos é muito mais constante e o crescimento do tempo é muito linear com o tempo de execução.

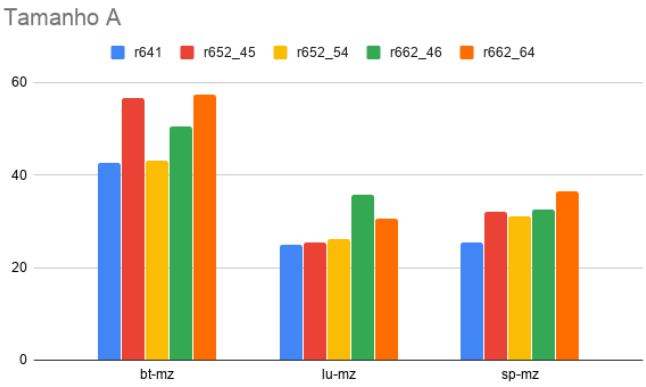
Algo que verificamos constantemente foi a diferença causada pela alteração das inner e outter threads.

2.1.7 Resultados Obtidos (Used Memory)

Depois de analisados os resultados de memória utilizada mais uma vez conseguimos concluir que os diferentes valores de threads são os que permitem a maior alteração dos resultados obtidos. Concluimos também que as



Tempo de Execução(sec) @ MPI, C

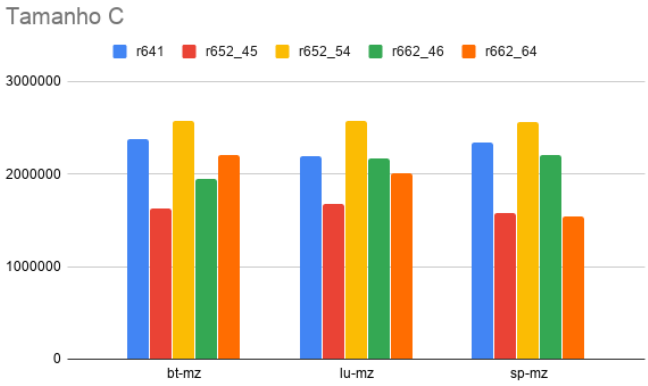


Tempo de Execução(sec) @ SER, A

máquinas 652 são as que apresentam melhores resultados quando otimizados, tendo os testes que executamos.

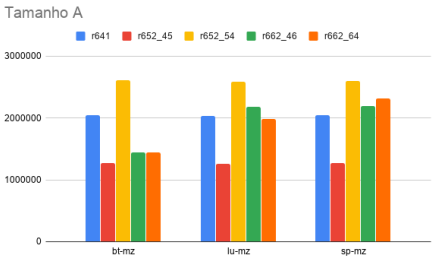
Em todas as versões reparamos que resultados entre as diferentes dimensões levou a um crescimento da memoria usada, mas um crescimento muito pouco significativo tendo em conta o aumento da dimensão dos dados, no entanto os resultados entre máquinas diferentes mantem-se constante.

Conseguimos observar nestes dois gráficos que na máquina 652 as diferentes dimensões de inner e outter threads são as que provocam uma maior alteração dos resultados, neste caso temos um crescimento de mais de 100% de um para o outro em quase todas as versões e dimensão dos dados. Já na máquina 662 a alteração não é tão significativa com apenas 5% de diferença na maior parte das vezes. Os testes que possuímos não são exaustivos suficiente para percebermos se as limitações são do algoritmo ou do hardware, mas tendo em conta os resultados de tempo que analisamos antes,

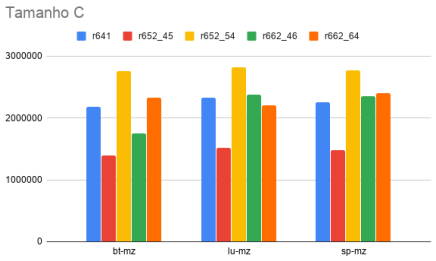


Used Memory @ MPI, C

calculamos que seja por causa do crescimento do algoritmo.



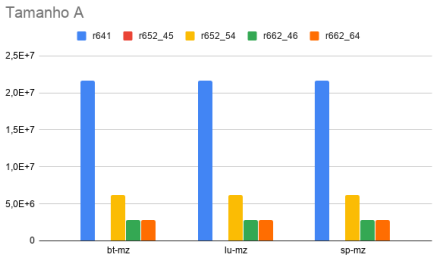
Used Memory @ SER, A



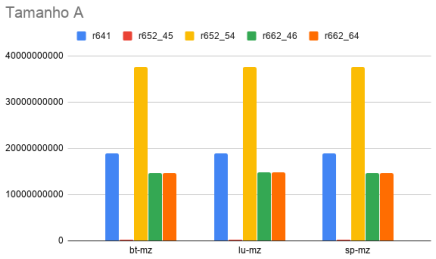
Used Memory @ SER, C

IO-WAIT cpu ticks e IDLE cpu ticks

Duas das metricas que tentamos também analizar foi a quantidade de cpu ticks que esteve em io-wait e em idle, verificamos aqui que esta informação é demasiado constante independentemente do tamanho, do benchmark e do algoritmo. Deixamos em anexo os diferentes gráficos gerados dos dados retiramos sobre os quais acabamos por não retirar conclusões por parecerem ser dados sem informação.



idle cpu ticks(biliões) @ SER, A



idle cpu ticks(biliões) @ SER, A

2.2 Utilitários de Monitorização

O objetivo desta fase é testar vários comandos de kernel de forma a nos apercebermos o que podemos obter apartir dos mesmos. Os tamanhos que iremos testar serão de A a F. Para a esta fase do trabalho utilizou-se a máquina r662 e testou-se o problema SP-ZE pois foi o que obteve menos problemas em termos de compilação e execução dos testes anteriores. Estes foram os comandos utilizados para os testes e consequentes resultados obtidos:

2.2.1 vmstat

O vmstat é uma ferramenta de monitorização que mostra informação acerca de memória, processos, interrupções, paginação e bloqueamento por I/O.

Usamos esta ferramenta principalmente para analisar como a utilização da memória é alterada.

```
vmstat -s
```

2.2.2 top

O top permite ao utilizador observar os processos em execução no sistema.

A utilização desta ferramenta permitiu analizar em tempo real a criação de processos e principalmente a utilização de CPU.

```
top -n 1 -b | grep "a77230"
```

2.2.3 ps

O ps é um comando utilizado para mostras os processos em execução no sistema. Serviu apenas para complementar a informação observada com o comando top.

```
ps
```

2.2.4 mpstat

O mpstat é uma ferramenta utilizada para relatar estatísticas relacionadas com o processador. Serviu também para complementarmos a informação do estado de utilização do CPU e outras métricas.

```
mpstat
```

2.2.5 lsof

O lsof é um comando utilizado para obter uma lista de todos os ficheiros abertos e respetivos processos que os abriram.

2.2.6 iostat

O iostat é uma ferramenta de monitorização que obter estatísticas acerca de entrada e saída no armazenamento do sistema operativo.

```
lsof -u a77230
```

2.3 Trabalho Futuro

Como trabalho futuro penso que seria importante experimentar diferentes tipos de compiladores e flags dos mesmos, além de também ser interessante experimentar com diferentes disposições entre threads interiores e exteriores ou threads e processos. As possibilidades são imensas e é isso que dá um grande valor à experimentação nestes meios.

Capítulo 3

TP2 - DTrace (Desenvolvimento de Programas)

3.1 Ex 1. Open Tracer

A chave para a implementação do Open Tracer baseia-se à volta de dois operadores:

- **?** - Representando o operador condicional devido às independências entre certas flags.
- **&** - Bitwise AND que permite a verificação das flags sendo que sabemos o bit correspondente a cada uma.

```
syscall::openat:entry
{
self->path = copyinstr(arg1);
self->flags = arg2;
}
```

Como se pode observar, no probe de entrada para a função openat, guarda-se o path e as flags, as quais se encontram no terceiro argumento da função, de forma a serem utilizadas quando o probe de retorno é ativado.

```
syscall::openat:return
{
this->return_out = arg0 == -1 ? "UNSUCCESSFUL" : "SUCCESSFUL";

this->flags_out =
    strjoin(self->flags & O_WRONLY ? "O_WRONLY"
```

```
                        : self->flags & O_RDWR ? "O_RDWR"
                                               : "O_RDONLY",
            strjoin(self->flags & O_APPEND ? "|O_APPEND"
                                           : "",
                    self->flags & O_CREAT ? "|O_CREAT"
                                          : ""));

    printf("%6d %6d %6d %15s %20s %70s\n",
           pid, uid, gid, this->return_out,
           this->flags_out, self->path);
}
```

No probe de retorno, em primeiro lugar testa-se se o comando foi executado com sucesso ou não. De seguida testa-se a existência ou não das flags com bitwise and, e condicionais, sendo que a existência de certas flags é dependente da inexistência de outras, como por exemplo O_WRONLY, O_RDWR e O_RDONLY. Por outro lado O_APPEND e O_CREAT são completamente independentes das outras flags.

No final de cada probe de retorno imprime-se no ecrã o **pid** (Process ID), **uid** (User ID), **gid** (Group ID), **return_out** (Sucesso ou insucesso da execução), **flags_out** (String correspondente ao estado das flags) e **path** (Path correspondente ao ficheiro que está a ser aberto)

É de notar que se as variáveis path e flags têm um prefixo ->self de forma a manter essas vaiáveis no mesmo thread e não só no probe em si.

3.2 Ex 2. Process Open Tracer

Utilizando algum do funcionamento do programa anterior, agora queremos obter, de tempo em tempo, informação acerca do process id e do nome de comando, particularmente o número de vezes que o um processo tentou abrir, criar e abrir com sucesso um certo ficheiro e retornar estas estatísticas.

Para este efeito utilizaram-se agregações de forma a registar eficientemente e de fácil acesso a informação que desejamos.

```
syscall::openat:entry
{
    self->flags = arg2;
    @opens[pid,execname] = count();
}
```

No probe de entrada guarda-se então a informação acerca das flags para ser utilizada mais tarde. Também se guarda numa agregação o número de opens realizados utilizando como chave tanto o pid (Process ID) e o execname (Nome do programa).

```
syscall::openat:return
{
    this->create = self->flags & O_CREAT ? 1 : 0;
    @creates[pid,execname] = sum(this->create);

    this->successful = arg0 == -1 ? 0 : 1;
    @successful[pid,execname] = sum(this->successful);
}
```

No probe de retorno testa-se tanto se a flag O_CREAT se encontra ativa como se o programa executou corretamente ou não atribuindo-lhes um valor de 0 ou 1. Seguidamente somam-se os resultados obtidos em mais duas agregações, de forma a serem utilizadas no fim do programa.

```
tick-$1sec
{
    printf("%-20Y\n",walltimestamp);
    printa("%6d %30s %@6d %@6d %@12d\n",
            @opens,@creates,@successful);
}
```

Por último, a cada 1 segundo é impressa a timestamp corrente e os dados referentes aos opens, crates e successful referente a cada processo e nome do programa.

3.3 Ex 3. Custom Truss

Para a construção de um script que permita um funcionamento semelhante a um comando truss utilizaram-se as seguintes probes:

```
syscall:::entry
/execname == $$1/
{
    @num[probefunc] = count();
    self->start_time = timestamp;
}
```

Na probe de entrada utiliza-se como condição o execname ser igual ao primeiro argumento do script, garantindo assim apenas system calls provenientes do mesmo.

De seguida, guarda-se numa agregação, utilizando como chave a função que foi chamada, o número de vezes que a mesma foi chamada, e também guardando o timestamp do começo dessa função em particular.

```
syscall:::return
/execname == $$1 && self->start_time != 0/
{
    @time[probefunc] = sum(timestamp-self->start_time);
}
```

Para a saída do probe além de se testar o execname, também se tem de garantir que o tempo de início é diferente de 0, ou seja a probe não foi chamada antes de o script iniciar.

Desta maneira agrega-se o tempo despendido na execução da função, de forma a somar todo o tempo gasto por esta função em particular.

```
dtrace:::END
{
    printa("%-20s %@10d %@10d\n",@num,@time);
}
```

Por fim imprime-se no ecrã os resultados correspondentes às agregações realizadas anteriormente.

3.4 Exemplos de Output

3.4.1 Exercício 1

Como não tinha acesso no ambiente Solaris à pasta /tmp criei uma pasta tmp no meu diretório de utilizador.

> cat /etc/inittab > tmp/test

```
  PID    UID    GID          RETURN                FLAGS
  PATH
24891   1010   5000      SUCCESSFUL     O_WRONLY|O_CREAT
tmp/test
24891   1010   5000    UNSUCCESSFUL             O_RDONLY
/var/ld/64/ld.config
24891   1010   5000      SUCCESSFUL             O_RDONLY
/lib/64/libc.so.1
24891   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/en_US.UTF-8
24891   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/common/amd64/methods_unicode.so.3
24891   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/pt_PT.UTF-8/pt_PT.UTF-8
24891   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
24891   1010   5000    UNSUCCESSFUL             O_RDONLY
```

```
 /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_lib_libc.mo
 24891   1010   5000      SUCCESSFUL             O_RDONLY
 /etc/inittab
```

> cat /etc/inittab >> tmp/test

```
   PID    UID    GID          RETURN                FLAGS
   PATH
24947   1010   5000      SUCCESSFUL O_WRONLY|O_APPEND|O_CREAT
 tmp/test
 24947   1010   5000    UNSUCCESSFUL             O_RDONLY
 /var/ld/64/ld.config
 24947   1010   5000      SUCCESSFUL             O_RDONLY
 /lib/64/libc.so.1
 24947   1010   5000      SUCCESSFUL             O_RDONLY
 /usr/lib/locale/en_US.UTF-8/en_US.UTF-8
 24947   1010   5000      SUCCESSFUL             O_RDONLY
 /usr/lib/locale/common/amd64/methods_unicode.so.3
 24947   1010   5000    UNSUCCESSFUL             O_RDONLY
 /usr/lib/locale/pt_PT.UTF-8/pt_PT.UTF-8
 24947   1010   5000    UNSUCCESSFUL             O_RDONLY
 /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
 24947   1010   5000    UNSUCCESSFUL             O_RDONLY
 /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_lib_libc.mo
```

> cat /etc/inittab | tee tmp/test

```
   PID    UID    GID          RETURN                FLAGS
  PATH
24986   1010   5000    UNSUCCESSFUL             O_RDONLY
/var/ld/64/ld.config
 24986   1010   5000      SUCCESSFUL             O_RDONLY
 /lib/64/libc.so.1
 24986   1010   5000      SUCCESSFUL             O_RDONLY
 /usr/lib/locale/en_US.UTF-8/en_US.UTF-8
 24986   1010   5000      SUCCESSFUL             O_RDONLY
 /usr/lib/locale/common/amd64/methods_unicode.so.3
 24986   1010   5000    UNSUCCESSFUL             O_RDONLY
 /usr/lib/locale/pt_PT.UTF-8/pt_PT.UTF-8
 24986   1010   5000    UNSUCCESSFUL             O_RDONLY
 /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
 24986   1010   5000    UNSUCCESSFUL             O_RDONLY
 /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_lib_libc.mo
 24986   1010   5000      SUCCESSFUL             O_RDONLY
```

```
/etc/inittab
24987   1010   5000    UNSUCCESSFUL             O_RDONLY
/var/ld/64/ld.config
24987   1010   5000      SUCCESSFUL             O_RDONLY
/lib/64/libc.so.1
24987   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/en_US.UTF-8
24987   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/common/amd64/methods_unicode.so.3
24987   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/pt_PT.UTF-8/pt_PT.UTF-8
24987   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
24987   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_lib_libc.mo
24987   1010   5000      SUCCESSFUL     O_WRONLY|O_CREAT
tmp/test
```

> cat /etc/inittab | tee -a tmp/test

```
  PID    UID    GID          RETURN                FLAGS
  PATH
25023   1010   5000    UNSUCCESSFUL             O_RDONLY
/var/ld/64/ld.config
25023   1010   5000      SUCCESSFUL             O_RDONLY
/lib/64/libc.so.1
25023   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/en_US.UTF-8
25023   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/common/amd64/methods_unicode.so.3
25023   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/pt_PT.UTF-8/pt_PT.UTF-8
25023   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
25023   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_lib_libc.mo
25023   1010   5000      SUCCESSFUL O_WRONLY|O_APPEND|O_CREAT
tmp/test
25022   1010   5000    UNSUCCESSFUL             O_RDONLY
/var/ld/64/ld.config
25022   1010   5000      SUCCESSFUL             O_RDONLY
/lib/64/libc.so.1
25022   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/en_US.UTF-8
```

```
25022   1010   5000      SUCCESSFUL             O_RDONLY
/usr/lib/locale/common/amd64/methods_unicode.so.3
25022   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/pt_PT.UTF-8/pt_PT.UTF-8
25022   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
25022   1010   5000    UNSUCCESSFUL             O_RDONLY
/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_lib_libc.mo
25022   1010   5000      SUCCESSFUL             O_RDONLY
/etc/inittab
```

3.4.2 Exercício 2

```
a77230@solaris:~/TP2$ ./openattrace2.d
##########################################
#              Openat Tracer             #
##########################################
   PID                           CMD    OPEN CREATE  SUCCESSFUL
2020 Jun  9 13:07:10
   911                        sstored     46     28          46
2020 Jun  9 13:07:11
   804                            fmd      1      0           0
 24405                  openattrace2.d      2      0           2
   911                        sstored     62     38          62
2020 Jun  9 13:07:12
   804                            fmd      1      0           0
 24405                  openattrace2.d      2      0           2
   911                        sstored     74     44          74
2020 Jun  9 13:07:13
   804                            fmd      1      0           0
 24405                  openattrace2.d      2      0           2
   911                        sstored     94     54          94
2020 Jun  9 13:07:14
   804                            fmd      1      0           0
 24405                  openattrace2.d      2      0           2
   911                        sstored     94     54          94
```

3.4.3 Exercício 3

```
a77230@solaris:~/TP2$ ./customstrace.d ls
##############################################
#               Custom STrace                #
##############################################
System Call                    #   Time(ns)
^C
rexit                          1          0
getpid                         1       3550
sysconfig                      1       3853
lwp_private                    1       4598
sigpending                     1       5946
getrlimit                      1       6341
systeminfo                     1      14805
write                          1      33168
setcontext                     2      15489
getdents                       2      45198
mmapobj                        2     146882
ioctl                          3      33596
mmap                           3      37978
close                          4      21147
```

resolvepath	4	132887
brk	5	33466
memcntl	5	85453
fstatat	7	90773
openat	8	261661

3.5 Possíveis melhorias

No primeiro exercício obtem-se um problema por vezes em que o probe não consegue identificar o argumento da função correspondente às flags e retorna um erro.

3.6 Conclusão

Após terminar este trabalho, conseguimos perceber melhor como é possível compreender o comportamento de cada algoritmo sem ver o código fonte. A utilização destas ferramentas pertimite-nos monitorizar todo o sistema e perceber o que está acontecer e analisar essa informação. Isto possibilita-nos, perceber melhor o que cada programa faz e no que isso afeta o sistema.

Dada a quantidade e variedade de testes que tivemos que realizar, utilizamos uma variedade de scripts que realizassem os testes no cluster SeARCH, com vista ao registo das diversas métricas. Além do conjunto de competências inerentes à disciplina, este trabalho permitiu-nos agilizar conhecimentos sobre scripting, profiling, sobre o sistema PBS e tratamento de informacão com recurso a diversos comandos unix.

Capítulo 4

TP3 - Implementação e análise de programas com recurso à ferramenta DTrace

4.1 Implementação com PThreads

Na implementação com pthreads utilizou-se uma barreira tal como na versão original openmp na qual cada thread fica com um bloco de linhas da matriz, correspondente à divisão do número de linhas pelo número de threads.

Retirou-se os cálculos matriciais da função main, de forma a que estes possam ser atribuídos como função a cada thread.

4.2 Implementação com C++11

Uma das dificuldades da implementação em C++11 é o facto de a biblioteca threads não conter nenhuma implementação de barreira, tendo sido apenas implementada na versão C++20, e como tal tive de escrever a minha própria implementação de uma barreira com mutexes.

```
class Barrier {
public:
    explicit Barrier(std::size_t iCount) :
      mThreshold(iCount),
      mCount(iCount),
      mGeneration(0) {
    }

    void Wait() {
        std::unique_lock<std::mutex> lLock{mMutex};
        auto lGen = mGeneration;
```

```
        if (!--mCount) {
            mGeneration++;
            mCount = mThreshold;
            mCond.notify_all();
        } else {
            mCond.wait(lLock, [this, lGen] {
                return lGen != mGeneration;
            });
        }
    }

private:
    std::mutex mMutex;
    std::condition_variable mCond;
    std::size_t mThreshold;
    std::size_t mCount;
    std::size_t mGeneration;
};
```

4.3 Pontas de proba USDT

De forma a marcar fins de iterações e obter informações sobre os programas a serem utilizados, foram implementadas algumas prontas de proba definidas pelo utilizador, as quais foram incluídas no programa:

- matrix_generation(int): Retornada quando se inicia a geração das matrizes. Retorna o tamanho das matrizes que serão usadas.
- start_calc(): Retornada quando se acaba a geração e alocação das matrizes e quando se iniciam os cálculos em si.
- start_iteration(): Retornada quando se inicia uma iteração do cálculo da dispersão de calor.
- start_copy(): Retornada quando se acabam os cálculos da dispersão de calor e se está a copiar os dados de uma matriz para a outra de forma a se prosseguir para a próxima iteração.
- end_iteration(int): Retornada no fim de cada iteração. Como resultado de retorno é enviada a iteração corrente.
- end_calc(): Retornada no fim de todas as iterações do programa.

De forma a garantir que não são retornadas mais pontas de proba USDT do que o ideal, definiu-se que apenas o master thread (o thread com ranque

0) poderá ativar a ponta de proba. Esta ativação apenas será realizada diretamente a seguir às barreiras implementadas, de tal forma que exista o mínimo de disparidade entre os resultados.

4.4 D script

Para interpretação das pontas de proba foi escrito um D script que gera texto baseado nas pontas de proba descritas acima, além de outras provenientes dos diversos providers.

Para todas as pontas de proba que não são USDT colocou-se uma cláusula referente ao execname do programa, para garantir que a ponta de proba é ativada pelo mesmo, como por exemplo:

```
/execname = "pthreads"/
```

Em geral, nas probes USDT definidas guarda-se o tempo ao qual elas foram disparadas de forma a se poder gerar médias, máximos e mínimos em relação a tempos de iteração, por exemplo.

Em relação a sondas pré-definidas:

- **syscall::open*:entry** - É impresso o caminho do ficheiro que foi aberto pela aplicação. É utilizado normlamente para observar o caminho do ficheiro no qual se guarda o resultado da matriz final.
- **syscall::pwrite*:entry** - É impresso o caminho do ficheiro no qual se começou a escrever.
- **syscall::pwrite*:return** - É impresso o caminho do ficheiro quando este é fechado.
- **sched:::on-cpu** - Imprime quando um thread começou a correr.
- **sched:::off-cpu** - Imprime quando um thread parou de correr.
- **sched:::sleep** - Guarda numa estrutura o tempo em que cada thread adormece.
- **sched:::wakeup** - Utiliza o valor obtido anteriormente por sched:::sleep de forma a medir quanto tempo os threads estiveram a dormir.
- **lockstat:::adaptive-block** - Guarda quantas vezes os threads foram bloqueados por barreiras etc.
- **proc:::exec** - É impresso o pid de um processo quando este inicia a sua execução.
- **proc:::exec-failure** - É impresso o pid de um processo quando este termina sem sucesso a sua execução.

- **proc:::exec-success** - É impresso o pid de um processo quando este termina com sucesso a sua execução.

A sonda dtrace:::END imprime no final da execução do programa os dados referentes à execução do programa, os quais foram guardados pelas sondas anteriormente. Estes resultados podem ser vistos na seguinte probe:

Em relação às USDT probes, estas são utilizadas para obter estatísticas em relação a tempos de execução e de bloqueio, etc., os quais irão ser explicados seguidamente.

Inicialmente definiram-se as seguintes variáveis:

- uint64$_t m_size$ − *Serve para guardar o tamanho de matriz, o qual é recebido pela probe query*$_m atrix_g en$ *Guarda o tempo de geração das matrizes.*

- uint64$_t alg_time$ − *Guarda o tempo que o algoritmo de dissipação de calor em si demorou a correr.* uint64$_t slee$ *Guarda o tempo que cada thread esteve a dormecido.*

- self int iteration$_s tart$ − *Guarda o tempo de início de uma iteração.* $self int copy_s tart$ − *Guarda o tempo de iní*

- this int it$_t ime$ − *Guarda o tempo de cada iteração (para ser usado em aggregates).* $this int copy_t ime$ − *Guarda o tempo de cópia de uma matriz para a outra dentro de uma iteração (para ser usado em aggregates).*

- this int calc$_t ime$ − *Guarda o tempo de cálculo de uma iteração (para ser usado em aggregates).*

```
heattimer*:::query-matrix_generation
{
    m_gen_time = timestamp;
    m_size = arg0;
}
```

Esta probe é lançada antes de se começar a alocar espaço em memória para as matrizes. de forma a obter o tempo inicial da geração de matrizes.

```
heattimer*:::query-start_calc
{
    m_gen_time = timestamp - m_gen_time;
    alg_time = timestamp;
}
```

Esta probe é lançada no fim da alocação de memória para as matrizes e antes de os calculos inicializarem, e serve para calcular o tempo de geração de matrizes e atribui o valor inicial do tempo que o algoritmo em si demorou a correr.

```
heattimer*:::query-start_iteration
{
    self->iteration_start = timestamp;
}
```

Esta probe serve apenas para marcar o tempo de início de uma iteração.

```
heattimer*:::query-start_copy
{
    self->copy_start = timestamp;
}
```

Esta probe serve apenas para marcar o tempo de início de uma cópia de matriz para matriz.

```
heattimer*:::query-end_iteration
{
    this->it_time = timestamp - self->iteration_start;
    this->copy_time = timestamp - self->copy_start;
    this->calc_time = this->it_time - this->copy_time;
    /*printf("Iteration %d finished on PROCESS: %d, THREAD: %d\n\tTime spent on calc
           arg0,
       pid,
           tid,
           this->calc_time,
           this->copy_time,
           this->it_time);*/

    @avg_calc_time = avg(this->calc_time);
    @max_calc_time = max(this->calc_time);
    @avg_copy_time = avg(this->copy_time);
    @max_copy_time = max(this->copy_time);
    @avg_it_time = avg(this->it_time);
    @max_it_time = max(this->it_time);
}

heattimer*:::query-end_calc
{
    printf("Program stopped calculating\n");
    alg_time = timestamp - alg_time;
}

dtrace:::END
{
printf("---------------------- Final Report ----------------------\n");
```

```
printf("Generated Matrices with size:           %dx%d\n",m_size,m_size);
printf("Time spent generating matrices:         %d\n",m_gen_time);
printf("Time spent running the main algorithm:  %d\n",alg_time);
printf("Iteration time:\n");
printa("    Average:                            %@d\n",@avg_it_time);
printa("    Maximum:                            %@d\n",@max_it_time);
printf("Calculation time:\n");
printa("    Average:                            %@d\n",@avg_calc_time);
printa("    Maximum:                            %@d\n",@max_calc_time);
printf("Copying time:\n");
printa("    Average:                            %@d\n",@avg_copy_time);
printa("    Maximum:                            %@d\n",@max_copy_time);
printa("Total number of threads locked:         %@d\n",@blocks);
printa("Time spent sleeping by thread %d        %@d\n",@sleep);
}
```

4.5 Resultados Obtidos

Como nota antes de mostrar os resultados obtidos é de salientar que a máquina Solaris disponibilizada foi partilhada pelos vários alunos e desta maneira os resultados obtidos em relação a tempos de execução não são necessáriamente corretos ou viáveis.

4.5.1 Implementação sequencial

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Press ENTER to start the program: Program stopped calculating
Opened the matrix file: /dev/dtrace/helper
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          58667157
Time spent running the main algorithm:   4669361532
Iteration time:
    Average:                             4646320
    Maximum:                             23738109
Calculation time:
    Average:                             3341146
    Maximum:                             21251183
Copying time:
    Average:                             1305173
    Maximum:                             2486926
Time spent sleeping by thread 1         5985298640
```

4.5.2 Implementação OpenMP

2 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Press ENTER to start the program: Time running: 3.374516
Program stopped calculating
Opened the matrix file: /dev/dtrace/helper
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          266562042
Time spent running the main algorithm:   3374472491
Iteration time:
    Average:                             3203098
    Maximum:                             153854472
Calculation time:
    Average:                             2123330
    Maximum:                             99820044
Copying time:
    Average:                             1079768
    Maximum:                             151743270
Total number of threads locked:          7
Time spent sleeping by thread 2         1499299485
Time spent sleeping by thread 1         3748823173
```

4 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Program stopped calculating
Press ENTER to start the program: Time running: 3.628665
Opened the matrix file: /dev/dtrace/helper
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          25847052
Time spent running the main algorithm:   3628511571
Iteration time:
    Average:                             2605295
    Maximum:                             111157740
Calculation time:
    Average:                             1802231
    Maximum:                             109995746
```

```
Copying time:
    Average:                              803063
    Maximum:                              38741696
Total number of threads locked:           18
Time spent sleeping by thread 1          2115218981
Time spent sleeping by thread 3          2181612902
Time spent sleeping by thread 2          2284874073
Time spent sleeping by thread 4          2322364484
```

8 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Program stopped calculating
Press ENTER to start the program: Time running: 2.840380
Opened the matrix file: /dev/dtrace/helper
---------------------- Final Report ----------------------
Generated Matrices with size:             1024x1024
Time spent generating matrices:           80519955
Time spent running the main algorithm:    2840337993
Iteration time:
    Average:                              1843545
    Maximum:                              21369525
Calculation time:
    Average:                              1062121
    Maximum:                              11639497
Copying time:
    Average:                              781423
    Maximum:                              18660511
Total number of threads locked:           15
Time spent sleeping by thread 7          1972132727
Time spent sleeping by thread 3          1984181907
Time spent sleeping by thread 6          1991912581
Time spent sleeping by thread 2          1992744452
Time spent sleeping by thread 1          2015936313
Time spent sleeping by thread 8          2023470676
Time spent sleeping by thread 5          2069151421
Time spent sleeping by thread 4          2267338987
```

4.5.3 Implementação MPI

Infelizmente não me foi possível correr a versão MPI com sucessos, pois estava a obter um erro que não consegui resolver a tempo da entrega do relatório.

4.5.4 Implementação com PThreads

2 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Press ENTER to start the program: Opened the matrix file: /dev/dtrace/helper
Program stopped calculating
Opened the matrix file: /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          56728854
Time spent running the main algorithm:   2601083594
Iteration time:
    Average:                             2583421
    Maximum:                             47259391
Calculation time:
    Average:                             1836347
    Maximum:                             45679083
Copying time:
    Average:                             747074
    Maximum:                             2292989
Total number of threads locked:          1
Time spent sleeping by thread 3         631840690
Time spent sleeping by thread 2         982669178
Time spent sleeping by thread 1         9702239149
```

4 threads

```
Tracer is ready!
Opened the matrix file: result.txt
Opened the matrix file: /dev/dtrace/helper
Press ENTER to start the program: Opened the matrix file: /usr/lib/locale/en_US.UTF-
Opened the matrix file: /dev/dtrace/helper
Program stopped calculating
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          135345753
Time spent running the main algorithm:   2377443323
Iteration time:
    Average:                             2360002
    Maximum:                             137274737
Calculation time:
    Average:                             1423624
```

```
    Maximum:                              112654588
Copying time:
    Average:                              936377
    Maximum:                              50734609
Total number of threads locked:           3
Time spent sleeping by thread 1           66865
Time spent sleeping by thread 5           495512152
Time spent sleeping by thread 4           562690185
Time spent sleeping by thread 3           581875649
Time spent sleeping by thread 2           745362976
```

8 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Press ENTER to start the program: Opened the matrix file: /usr/lib/locale/en_US.UTF-
Opened the matrix file: /dev/dtrace/helper
Program stopped calculating
---------------------- Final Report ----------------------
Generated Matrices with size:             1024x1024
Time spent generating matrices:           80505414
Time spent running the main algorithm:    2804115343
Iteration time:
    Average:                              2779589
    Maximum:                              162960841
Calculation time:
    Average:                              1413562
    Maximum:                              123956787
Copying time:
    Average:                              1366026
    Maximum:                              162335914
Total number of threads locked:           5
Time spent sleeping by thread 8           320962189
Time spent sleeping by thread 9           448724426
Time spent sleeping by thread 5           538287898
Time spent sleeping by thread 2           549681876
Time spent sleeping by thread 4           575055364
Time spent sleeping by thread 3           627603164
Time spent sleeping by thread 6           713475331
Time spent sleeping by thread 7           921782004
Time spent sleeping by thread 1           3267530171
```

4.5.5 Implementação em C++11

2 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Press ENTER to start the program: Program stopped calculating
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          332075828
Time spent running the main algorithm:   2984016980
Iteration time:
    Average:                             2949100
    Maximum:                             156551851
Calculation time:
    Average:                             1891426
    Maximum:                             33125169
Copying time:
    Average:                             1057673
    Maximum:                             153113578
Total number of threads locked:          4
Time spent sleeping by thread 3         821964159
Time spent sleeping by thread 2         1135193402
Time spent sleeping by thread 1         4299053232
```

4 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Program stopped calculating
Opened the matrix file: /usr/lib/locale/en_US.UTF-8/LC_MESSAGES/solaris_linkers.mo
Press ENTER to start the program: Opened the matrix file: /dev/dtrace/helper
---------------------- Final Report ----------------------
Generated Matrices with size:            1024x1024
Time spent generating matrices:          46885006
Time spent running the main algorithm:   2744923028
Iteration time:
    Average:                             2728525
    Maximum:                             194461527
Calculation time:
    Average:                             1885908
```

```
    Maximum:                                193656004
Copying time:
    Average:                                842616
    Maximum:                                101989370
Total number of threads locked:             5
Time spent sleeping by thread 1             46001
Time spent sleeping by thread 5             859894303
Time spent sleeping by thread 3             1122221533
Time spent sleeping by thread 4             1193721878
Time spent sleeping by thread 2             1359281778
```

8 threads

```
Tracer is ready!
Opened the matrix file: /dev/dtrace/helper
Opened the matrix file: result.txt
Press ENTER to start the program: Opened the matrix file: /usr/lib/locale/en_US.UTF-
Opened the matrix file: /dev/dtrace/helper
Program stopped calculating
---------------------- Final Report ----------------------
Generated Matrices with size:               1024x1024
Time spent generating matrices:             34491684
Time spent running the main algorithm:      2541628137
Iteration time:
    Average:                                2520711
    Maximum:                                152327165
Calculation time:
    Average:                                1192145
    Maximum:                                80788941
Copying time:
    Average:                                1328566
    Maximum:                                151038845
Total number of threads locked:             13
Time spent sleeping by thread 1             246946368
Time spent sleeping by thread 3             1468760810
Time spent sleeping by thread 8             1549090749
Time spent sleeping by thread 7             1610508844
Time spent sleeping by thread 4             1616914297
Time spent sleeping by thread 5             1643953608
Time spent sleeping by thread 6             1665085987
Time spent sleeping by thread 2             1768867876
Time spent sleeping by thread 9             1776747141
```

4.6 Anexos

4.6.1 Implementação sequencial

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "heattimer.h"

#define N_MAX 1000
#define MAT_SIZE 1024
#define M_SIZE MAT_SIZE + 2

int main()
{

    printf("Press ENTER to start the program: ");
    scanf("*");

    FILE *file = fopen("result.txt", "w+");

    if(HEATTIMER_QUERY_MATRIX_GENERATION_ENABLED())
        HEATTIMER_QUERY_MATRIX_GENERATION(MAT_SIZE);

    int **G1, **G2;

    G1 = (int **)malloc(sizeof(int *) * M_SIZE);
    G2 = (int **)malloc(sizeof(int *) * M_SIZE);

    for (int i = 0; i < M_SIZE; i++)
    {
        G1[i] = (int *)malloc(sizeof(int) * M_SIZE);
        G2[i] = (int *)malloc(sizeof(int) * M_SIZE);
    }

    for (int i = 0; i < M_SIZE; i++)
    {
        for (int j = 0; j < M_SIZE; j++)
        {
            G1[i][j] = 0;
            G2[i][j] = 0;
        }
    }
```

```
//Filling the lower line of the matrix with the highest heat
for (int i = 0; i < M_SIZE; i++)
{
    G1[i][0] = 0xffffff; //Hexcode ffffff
}

if(HEATTIMER_QUERY_START_CALC_ENABLED())
    HEATTIMER_QUERY_START_CALC();

for (int it = 0; it < N_MAX; it++)
{
    if(HEATTIMER_QUERY_START_ITERATION_ENABLED())
        HEATTIMER_QUERY_START_ITERATION();

    for (int i = 1; i < M_SIZE - 1; i++)
    {
        for (int j = 1; j < M_SIZE - 1; j++)
        {
            G2[i][j] = (G1[i - 1][j] + G1[i + 1][j] + G1[i][j - 1] +
        }
    }

    if(HEATTIMER_QUERY_START_COPY_ENABLED())
        HEATTIMER_QUERY_START_COPY();

    //Copiar G2 para G1
    for (int i = 1; i < M_SIZE - 1; i++)
    {
        for (int j = 1; j < M_SIZE - 1; j++)
        {
            G1[i][j] = G2[i][j];
        }
    }

    if(HEATTIMER_QUERY_END_ITERATION_ENABLED())
        HEATTIMER_QUERY_END_ITERATION(it);
}

if(HEATTIMER_QUERY_END_CALC_ENABLED())
    HEATTIMER_QUERY_END_CALC();

//Prints results to a file
for (int i = 0; i < M_SIZE; i++)
{
```

```
            for (int j = 0; j < M_SIZE; j++)
                fprintf(file, "%d|", G1[i][j]);
        }
        fprintf(file, "\n");
    }
```

4.6.2 Implementação OpenMP

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
#include "heattimer.h"

#define N_MAX 1000
#define N_THREADS 8
#define MAT_SIZE 1024
#define M_SIZE MAT_SIZE + 2

int main()
{
    printf("Press ENTER to start the program: ");
    scanf("*");

    FILE *file = fopen("result.txt", "w+");

    if(HEATTIMER_QUERY_MATRIX_GENERATION_ENABLED())
        HEATTIMER_QUERY_MATRIX_GENERATION(MAT_SIZE);

    int **G1, **G2;

    G1 = (int **)malloc(sizeof(int *) * M_SIZE);
    G2 = (int **)malloc(sizeof(int *) * M_SIZE);

    for (int i = 0; i < M_SIZE; i++)
    {
        G1[i] = (int *)malloc(sizeof(int) * M_SIZE);
        G2[i] = (int *)malloc(sizeof(int) * M_SIZE);
    }

    for (int i = 0; i < M_SIZE; i++)
    {
        for (int j = 0; j < M_SIZE; j++)
        {
            G1[i][j] = 0;
            G2[i][j] = 0;
        }
    }

    //Filling the lower line of the matrix with the highest heat
```

```
for (int i = 0; i < M_SIZE; i++)
{
    G1[i][0] = 0xffffff; //Hexcode ffffff
}

double start_time = omp_get_wtime();

omp_set_num_threads(N_THREADS);

if(HEATTIMER_QUERY_START_CALC_ENABLED())
    HEATTIMER_QUERY_START_CALC();

for (int it = 0; it < N_MAX; it++)
{

    #pragma omp parallel
    {
            #pragma omp master
        {
            if(HEATTIMER_QUERY_START_ITERATION_ENABLED())
                HEATTIMER_QUERY_START_ITERATION();
        }

        #pragma omp for schedule(static)
        for (int i = 1; i < M_SIZE - 1; i++)
        {
            for (int j = 1; j < M_SIZE - 1; j++)
            {
                G2[i][j] = (G1[i - 1][j] + G1[i + 1][j] + G1[i][j - 1
            }
        }

        #pragma omp master
        {
            if(HEATTIMER_QUERY_START_COPY_ENABLED())
                HEATTIMER_QUERY_START_COPY();
        }

        //Copiar G2 para G1
        #pragma omp for schedule(static)
        for (int i = 1; i < M_SIZE - 1; i++)
        {
            for (int j = 1; j < M_SIZE - 1; j++)
            {
```

```
                    G1[i][j] = G2[i][j];
                }
            }

            #pragma omp master
                {
                if(HEATTIMER_QUERY_END_ITERATION_ENABLED())
                    HEATTIMER_QUERY_END_ITERATION(it);
            }
        }
    }

    if(HEATTIMER_QUERY_END_CALC_ENABLED())
        HEATTIMER_QUERY_END_CALC();

    double end_time = omp_get_wtime();

    printf("Time running: %lf\n", end_time - start_time);

    //Prints results to a file
    for (int i = 0; i < M_SIZE; i++)
    {
        for (int j = 0; j < M_SIZE; j++)
            fprintf(file, "%d|", G1[i][j]);
    }
    fprintf(file, "\n");
}
```

4.6.3 Implentação MPI

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>
#include "heattimer.h"

#define N_MAX 1000
#define N_MACHINES 8
#define MAT_SIZE 1024
#define M_SIZE (MAT_SIZE + 2)

int main(int argc, char *argv[])
{
    printf("Press ENTER to start the program: ");
    scanf("*");

    FILE *file = fopen("result.txt", "w+");

    int rank;
    int i_division = MAT_SIZE / N_MACHINES;
    int MACH_MAT_SIZE = i_division * M_SIZE;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    double start_time = MPI_Wtime();
    int left_send_buffer[M_SIZE], right_send_buffer[M_SIZE], left_recv_bu
    int *final_send_buffer = (int *)malloc(sizeof(int) * M_SIZE * i_divis
    int *final_result_buffer = (int *)malloc(sizeof(int) * M_SIZE * MAT_S
    if (final_send_buffer == NULL || final_result_buffer == NULL)
    {
        printf("NULL, not enough memory\n");
    }
    MPI_Request left_send_request, right_send_request, left_recv_request,

    if(rank == 0){
        if(HEATTIMER_QUERY_MATRIX_GENERATION_ENABLED())
            HEATTIMER_QUERY_MATRIX_GENERATION(MAT_SIZE);
    }

    int **G1 = (int **)malloc(sizeof(int *) * i_division);
    int **G2 = (int **)malloc(sizeof(int *) * i_division);
    for (int i = 0; i < i_division; i++)
```

```
{
    G1[i] = (int *)malloc(sizeof(int) * M_SIZE);
    G2[i] = (int *)malloc(sizeof(int) * M_SIZE);

    for (int j = 0; j < M_SIZE; j++)
    {
        G1[i][j] = 0;
        G2[i][j] = 0;
    }

    G1[i][0] = 0xffffff; //Hexcode ffffff
    G2[i][0] = 0xffffff;
}

for (int j = 0; j < M_SIZE; j++)
{
    left_recv_buffer[j] = 0;
    right_recv_buffer[j] = 0;
}

if(rank == 0){
    if(HEATTIMER_QUERY_START_CALC_ENABLED())
        HEATTIMER_QUERY_START_CALC();
}
for (int it = 0; it < N_MAX; it++)
{

    if(HEATTIMER_QUERY_START_ITERATION_ENABLED())
        HEATTIMER_QUERY_START_ITERATION();

    //Computes the parts with no dependencies
    for (int i = 1; i < i_division - 1; i++)
    {
        for (int j = 1; j < M_SIZE - 1; j++)
        {
            G2[i][j] = (G1[i - 1][j] + G1[i + 1][j] + G1[i][j - 1] +
        }
    }

    //Waits to receive the left buffer
    if (it != 0 && rank != 0)
    {
        MPI_Wait(&left_recv_request, MPI_STATUS_IGNORE);
    }
```

```
for (int j = 1; j < M_SIZE - 1; j++)
{
    G2[0][j] = (left_recv_buffer[j] + G1[1][j] + G1[0][j - 1] + G
}

//Waits to receive the right buffer
if (it != 0 && rank != N_MACHINES - 1)
{
    MPI_Wait(&right_recv_request, MPI_STATUS_IGNORE);
}
for (int j = 1; j < M_SIZE - 1; j++)
{
    G2[i_division - 1][j] = (G1[i_division - 2][j] + right_recv_b
}

//Guarantees the buffers have been sent
if (it != 0)
{
    if (rank != 0)
    {
        MPI_Wait(&left_send_request, MPI_STATUS_IGNORE);
    }
    if (rank != N_MACHINES - 1)
    {
        MPI_Wait(&right_send_request, MPI_STATUS_IGNORE);
    }
}
//Copies the column to the buffer
for (int j = 0; j < M_SIZE; j++)
{
    if (rank != 0)
        left_send_buffer[j] = G1[0][j];
    if (rank != N_MACHINES - 1)
        right_send_buffer[j] = G1[i_division - 1][j];
}

//Sends and receives assynchonously the buffers
if (rank != 0)
{
    MPI_Isend(left_send_buffer, M_SIZE, MPI_INT, rank - 1, 0, MPI
    MPI_Irecv(left_recv_buffer, M_SIZE, MPI_INT, rank - 1, 0, MPI
}
if (rank != N_MACHINES - 1)
{
```

```
            MPI_Isend(right_send_buffer, M_SIZE, MPI_INT, rank + 1, 0, MP
            MPI_Irecv(right_recv_buffer, M_SIZE, MPI_INT, rank + 1, 0, MP
        }

        if(HEATTIMER_QUERY_START_COPY_ENABLED())
            HEATTIMER_QUERY_START_COPY();

        //Copies from G2 to G1
        for (int i = 0; i < i_division; i++)
        {
            for (int j = 0; j < M_SIZE; j++)
            {
                G1[i][j] = G2[i][j];
            }
        }

        if(HEATTIMER_QUERY_END_ITERATION_ENABLED())
            HEATTIMER_QUERY_END_ITERATION(it);
    }

    for (int i = 0; i < i_division; i++)
    {
        for (int j = 0; j < M_SIZE; j++)
        {
            final_send_buffer[i * M_SIZE + j] = G1[i][j];
        }
    }

    //Rank 0 gathers results from all other ranks
    //int MPI_Gather(void* sendbuf, int sendcount, MPI_Datatype sendtype,
    MPI_Gather(final_send_buffer, MACH_MAT_SIZE, MPI_INT, final_result_bu



    if (rank == 0)
    {
        if(HEATTIMER_QUERY_END_CALC_ENABLED())
            HEATTIMER_QUERY_END_CALC();

        //Creates the final matrix to output the result
        int **FINAL_MAT = (int **)malloc(sizeof(int *) * M_SIZE);
        for (int i = 0; i < M_SIZE; i++)
        {
            FINAL_MAT[i] = (int *)malloc(sizeof(int) * M_SIZE);
```

```
            }
            for (int j = 0; j < M_SIZE; j++)
            {
                FINAL_MAT[0][j] = 0;
                FINAL_MAT[M_SIZE - 1][j] = 0;
            }
            FINAL_MAT[0][0] = 0xffffff;
            FINAL_MAT[M_SIZE - 1][0] = 0xffffff;

            //Copies the result from the receive buffer to the result matrix
            for (int i = 0; i < MAT_SIZE; i++)
            {
                for (int j = 0; j < M_SIZE; j++)
                {
                    FINAL_MAT[i + 1][j] = final_result_buffer[i * M_SIZE + j]
                }
            }

            double end_time = MPI_Wtime();

            printf("Total time: %lf seconds\n", end_time - start_time);

            //Prints results to a file
            for (int i = 0; i < M_SIZE; i++)
            {
                for (int j = 0; j < M_SIZE; j++)
                {
                    fprintf(file, "%d|", FINAL_MAT[i][j]);
                }
                fprintf(file, "\n");
            }
        }

        MPI_Finalize();
    }
```

4.6.4 Implementação com PThreads

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>
#include <stdint.h>
#include "heattimer.h"

#define N_MAX 1000
#define MAT_SIZE 1024
#define M_SIZE MAT_SIZE + 2
#define N_THREADS 8
#define M_DIV MAT_SIZE / N_THREADS

int **G1, **G2;
pthread_barrier_t barrier;

void *heat_dispersion(void* tnum_void){
    int tnum = (intptr_t) tnum_void;
    for (int it = 0; it < N_MAX; it++)
    {

        if(tnum == 0)
            if(HEATTIMER_QUERY_START_ITERATION_ENABLED())
                HEATTIMER_QUERY_START_ITERATION();

        for (int i = (tnum * M_DIV) + 1; i < ((tnum + 1) * M_DIV) + 1; i+
        {
            for (int j = 1; j < M_SIZE - 1; j++)
            {
                G2[i][j] = (G1[i - 1][j] + G1[i + 1][j] + G1[i][j - 1] +
            }
        }

        pthread_barrier_wait(&barrier);

        if(tnum == 0)
            if(HEATTIMER_QUERY_START_COPY_ENABLED())
                    HEATTIMER_QUERY_START_COPY();

        for (int i = (tnum * M_DIV) + 1; i < ((tnum + 1) * M_DIV) + 1; i+
        {
            for (int j = 1; j < M_SIZE - 1; j++)
```

```
            {
                G1[i][j] = G2[i][j];
            }
        }

        pthread_barrier_wait(&barrier);

        if(tnum == 0)
            if(HEATTIMER_QUERY_END_ITERATION_ENABLED())
                HEATTIMER_QUERY_END_ITERATION(it);
    }
}

int main()
{

    printf("Press ENTER to start the program: ");
    scanf("*");

    FILE *file = fopen("result.txt", "w+");

    if(HEATTIMER_QUERY_MATRIX_GENERATION_ENABLED())
        HEATTIMER_QUERY_MATRIX_GENERATION(MAT_SIZE);

    G1 = (int **)malloc(sizeof(int *) * M_SIZE);
    G2 = (int **)malloc(sizeof(int *) * M_SIZE);

    for (int i = 0; i < M_SIZE; i++)
    {
        G1[i] = (int *)malloc(sizeof(int) * M_SIZE);
        G2[i] = (int *)malloc(sizeof(int) * M_SIZE);
    }

    for (int i = 0; i < M_SIZE; i++)
    {
        for (int j = 0; j < M_SIZE; j++)
        {
            G1[i][j] = 0;
            G2[i][j] = 0;
        }
    }

    for (int i = 0; i < M_SIZE; i++)
    {
```

```
            G1[i][0] = 0xffffff; //Hexcode ffffff
        }

        pthread_t* thread_handles = (pthread_t*) malloc(N_THREADS * sizeof(pt
        pthread_barrier_init(&barrier, (pthread_barrierattr_t*) NULL, N_THREA

        if(HEATTIMER_QUERY_START_CALC_ENABLED())
            HEATTIMER_QUERY_START_CALC();

        for(int thread = 0; thread < N_THREADS; thread++){
            pthread_create(&thread_handles[thread], (pthread_attr_t*) NULL,
                           heat_dispersion, (void*) (intptr_t) thread);
        }

        for(int thread = 0; thread < N_THREADS; thread++){
            pthread_join(thread_handles[thread],NULL);
        }

        free(thread_handles);
        pthread_barrier_destroy(&barrier);

        if(HEATTIMER_QUERY_END_CALC_ENABLED())
            HEATTIMER_QUERY_END_CALC();

        //Prints results to a file
        for (int i = 0; i < M_SIZE; i++)
        {
            for (int j = 0; j < M_SIZE; j++)
                fprintf(file, "%d|", G1[i][j]);
        }
        fprintf(file, "\n");
    }
```

4.6.5 Implementação em C++11

```
#include <iostream>
#include <thread>
#include <vector>
#include <mutex>
#include <condition_variable>
#include "heattimer.h"

#define N_MAX 1000
#define MAT_SIZE 1024
#define M_SIZE MAT_SIZE + 2
#define N_THREADS 8
#define M_DIV MAT_SIZE / N_THREADS

class Barrier {
public:
    explicit Barrier(std::size_t iCount) :
      mThreshold(iCount),
      mCount(iCount),
      mGeneration(0) {
    }

    void Wait() {
        std::unique_lock<std::mutex> lLock{mMutex};
        auto lGen = mGeneration;
        if (!--mCount) {
            mGeneration++;
            mCount = mThreshold;
            mCond.notify_all();
        } else {
            mCond.wait(lLock, [this, lGen]
            { return lGen != mGeneration; });
        }
    }

private:
    std::mutex mMutex;
    std::condition_variable mCond;
    std::size_t mThreshold;
    std::size_t mCount;
    std::size_t mGeneration;
};
```

```
void heat_dispersion(int tnum, int** G1, int** G2, Barrier *br){
    for (int it = 0; it < N_MAX; it++)
    {
        if(tnum == 0)
            if(HEATTIMER_QUERY_START_ITERATION_ENABLED())
                HEATTIMER_QUERY_START_ITERATION();

        for (int i = (tnum * M_DIV) + 1; i < ((tnum + 1) * M_DIV) + 1; i+
        {
            for (int j = 1; j < M_SIZE - 1; j++)
            {
                G2[i][j] = (G1[i - 1][j] + G1[i + 1][j] + G1[i][j - 1] +
            }
        }

        br->Wait();

        if(tnum == 0)
            if(HEATTIMER_QUERY_START_COPY_ENABLED())
                    HEATTIMER_QUERY_START_COPY();

        for (int i = (tnum * M_DIV) + 1; i < ((tnum + 1) * M_DIV) + 1; i+
        {
            for (int j = 1; j < M_SIZE - 1; j++)
            {
                G1[i][j] = G2[i][j];
            }
        }

        br->Wait();

        if(tnum == 0)
            if(HEATTIMER_QUERY_END_ITERATION_ENABLED())
                    HEATTIMER_QUERY_END_ITERATION(it);
    }
}

int main()
{
    char c{};
    std::cout << "Press ENTER to start the program: ";
    scanf("*");
```

```
FILE *file = fopen("result.txt", "w+");

if(HEATTIMER_QUERY_MATRIX_GENERATION_ENABLED())
    HEATTIMER_QUERY_MATRIX_GENERATION(MAT_SIZE);

int **G1, **G2;

G1 = (int **)malloc(sizeof(int *) * M_SIZE);
G2 = (int **)malloc(sizeof(int *) * M_SIZE);

for (int i = 0; i < M_SIZE; i++)
{
    G1[i] = (int *)malloc(sizeof(int) * M_SIZE);
    G2[i] = (int *)malloc(sizeof(int) * M_SIZE);
}

for (int i = 0; i < M_SIZE; i++)
{
    for (int j = 0; j < M_SIZE; j++)
    {
        G1[i][j] = 0;
        G2[i][j] = 0;
    }
}

//Filling the lower line of the matrix with the highest heat
for (int i = 0; i < M_SIZE; i++)
{
    G1[i][0] = 0xffffff; //Hexcode ffffff
}

std::thread threads[N_THREADS];

Barrier br(N_THREADS);

if(HEATTIMER_QUERY_START_CALC_ENABLED())
    HEATTIMER_QUERY_START_CALC();

for(int i = 0; i < N_THREADS; i++){
    threads[i] = std::thread(heat_dispersion,i,G1,G2,&br);
}

for(int i = 0; i < N_THREADS; i++){
    threads[i].join();
```

```
    }

    if(HEATTIMER_QUERY_END_CALC_ENABLED())
        HEATTIMER_QUERY_END_CALC();

    //Prints results to a file
    for (int i = 0; i < M_SIZE; i++)
    {
        for (int j = 0; j < M_SIZE; j++)
            fprintf(file, "%d|", G1[i][j]);
    }
    fprintf(file, "\n");
}
```

4.6.6 DScript (Exemplo para todos os programas com exceção da versão MPI)

```
#!/usr/sbin/dtrace -qs

uint64_t m_size;
uint64_t m_gen_time;
uint64_t alg_time;
uint64_t sleep_time[id_t];

self int iteration_start;
self int copy_start;
self string write_path;
self int asleep;

this int it_time;
this int copy_time;
this int calc_time;

dtrace:::BEGIN
{
    printf("Tracer is ready!\n");
}

heattimer*:::query-matrix_generation
{
    m_gen_time = timestamp;
    m_size = arg0;
}

heattimer*:::query-start_calc
{
    m_gen_time = timestamp - m_gen_time;
    alg_time = timestamp;
}

heattimer*:::query-start_iteration
{
    self->iteration_start = timestamp;
}

heattimer*:::query-start_copy
{
    self->copy_start = timestamp;
```

```
}

heattimer*:::query-end_iteration
{
    this->it_time = timestamp - self->iteration_start;
    this->copy_time = timestamp - self->copy_start;
    this->calc_time = this->it_time - this->copy_time;
    /*printf("Iteration %d finished on PROCESS: %d, THREAD: %d\n\tTime spent on calc
           arg0,
       pid,
           tid,
           this->calc_time,
           this->copy_time,
           this->it_time);*/

    @avg_calc_time = avg(this->calc_time);
    @max_calc_time = max(this->calc_time);
    @avg_copy_time = avg(this->copy_time);
    @max_copy_time = max(this->copy_time);
    @avg_it_time = avg(this->it_time);
    @max_it_time = max(this->it_time);
}

heattimer*:::query-end_calc
{
    printf("Program stopped calculating\n");
    alg_time = timestamp - alg_time;
}

syscall::open*:entry
/execname == "pthreads"/
{
    self->open_path = copyinstr(arg1);
    printf("Opened the matrix file: %s\n",self->open_path);
}

syscall::pwrite*:entry
/execname == "pthreads"/
{
    self->write_path = copyinstr(arg1);
    printf("Started writing in file: %s\n",self->write_path);
}

syscall::pwrite*:return
```

```
/execname == "pthreads"/
{
    printf("Finished writing in file: %s\n",self->write_path);
}

sched:::on-cpu
/execname == "pthreads"/
{
    /*printf("Thread %d started running\n",tid);*/
}

sched:::off-cpu
/execname == "pthreads"/
{
    /*printf("Thread %d stopped running\n",tid);*/
}

sched:::sleep
/execname == "pthreads"/
{
    sleep_time[tid] = timestamp;
}

sched:::wakeup
/execname == "pthreads" && sleep_time[tid] != 0/
{
    @sleep[tid] = sum(timestamp - sleep_time[tid]);
    sleep_time[tid] = 0;
}

lockstat:::adaptive-block
/execname == "pthreads"/
{
    @blocks = count();
}

proc:::exec
/execname == "pthreads"/
{
    printf("Process %d started executing\n",pid);
}

proc:::exec-failure
/execname == "pthreads"/
```

```
{
    printf("Process %d exectued unsuccessfully\n",pid);
}

proc:::exec-success
/execname == "pthreads"/
{
    printf("Process %d executed correctly\n",pid);
}


dtrace:::END
{
    printf("---------------------- Final Report ----------------------\n");
    printf("Generated Matrices with size:            %dx%d\n",m_size,m_size);
    printf("Time spent generating matrices:          %d\n",m_gen_time);
    printf("Time spent running the main algorithm:   %d\n",alg_time);
    printf("Iteration time:\n");
    printa("    Average:                             %@d\n",@avg_it_time);
    printa("    Maximum:                             %@d\n",@max_it_time);
    printf("Calculation time:\n");
    printa("    Average:                             %@d\n",@avg_calc_time);
    printa("    Maximum:                             %@d\n",@max_calc_time);
    printf("Copying time:\n");
    printa("    Average:                             %@d\n",@avg_copy_time);
    printa("    Maximum:                             %@d\n",@max_copy_time);
    printa("Total number of threads locked:          %@d\n",@blocks);
    printa("Time spent sleeping by thread %d         %@d\n",@sleep);
}
```

Capítulo 5

TP4 - Análise de aplicações com apoio à ferramenta perf

5.1 Parte 1 - Análise de Algoritmos de Sorting

O objetivo desta parte do trabalho prático passa por identificar e retirar informações acerca da maneira como diversos algoritmos de sorting se comportam, quais os algoritmos com melhor performance, etc.

Para esta fase utilizou-se o meu computador pessoal, o qual possúi um processador Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz. Como tal, especialmente nos FlameGraphs poder-se-á reparar na existência de outros processos a correr ao mesmo tempo que o programa que estamos a estudar.

5.1.1 Obtenção de valores

Para a obtenção de valores escolheram-se vários indicadores, neste caso: Instructions, Cycles e Cache-Misses.

Utilizando o comando:

```
perf stat -e cycles,instructions,cache-misses ./<sort> 1 //
1 100000000
```

Obtiveram-se os seguintes resultados:

Tipo de Sort	Cycles	Instructions	Cache-Misses	Tempo
Quick Sort	72,751 E+9	76,449 E+9	164,972 E+6	20,239 s
Radix Sort	46,950 E+9	81,785 E+9	446,865 E+6	13,400 s
Heap Sort	190,097 E+9	134,587 E+9	447,005 E+6	55,986 s
Merge Sort	113,383 E+9	176,448 E+9	408,424 E+6	35,511 s

Valores obtidos utilizando perf stat

Em relação à razão pela qual os algoritmos obtêm este valor, os algoritmos que obteram melhor performance foram, por esta ordem, Radix Sort > Quick Sort > Merge Sort > Heap Sort.

O Quick Sort é um algoritmo que apesar de não ter um valor muito elevado em relação a instruções por ciclo (1,05), possúi um tempo de execução bastante bom. Isto provávelmente deve-se ao facto da maneira como o algoritmo acede à memória, sendo que os acessos no início do algoritmo são bastante lentos, devido ao facto de este ter de aceder ao array inteiro, mas acelera à medida que o array é subdividido, o que acaba por gerar um número muito reduzido de cache-misses, cuja maior parte se encontra provávelmente no início da execução do programa. Por outro lado o desempenho do mesmo depende da aleatoriedade do conjunto e do quão bem é escolhido o pivot. Se a distribuição de valores não for uniforme, a escolha do pivot poderá não ser indicada para a resolução deste algoritmo, tornando assim o programa bastante mais lento.

O Radix Sort é um algoritmo cuja complexidade é de O(N), obtendo a melhor complexidade de todos os outros algoritmos aqui presentes. Isto faz com que apesar de o número de instruções e cache misses ser relativamente elevado, o facto de o programa não ter de realizar tantos ciclos como nos outros, e as comparações serem apenas dígito a dígito, faz com que as instruções sejam realizadas mais rápidamente.

O Heap Sort foi o algoritmo com os piores resultados, provávelmente pois este depende de alterar uma árvore de pesquisa, o que não é muito bom e termos de acessos à memória e localidade espacial, estas esperas de memória provávelmente foi o que causou um número de ciclos tão elevado, pois este tem de aceder a níveis superiores de cache.

O Merge Sort apesar de ter um número elevado de instruções por ciclo (1,56), e um número muito reduzido de cache-misses, possúi um grande número de instruções o que causa um tempo de execução mais alto do que o Quick Sort. Este elevado número de instruções pode ser causado pelas alocações de memória derivadas da geração do array auxiliar.

5.1.2 Perfil de Execução

Quick Sort

```
# Overhead       Samples  Command  Shared Object      Symbol
# ........  ............  .......  .................  ........................
#
    95.58%          1950  sort     sort               [.] sort1
     1.45%            25  sort     sort               [.] ini_vector
     1.09%            30  sort     sort               [.] copy_vector
     0.69%            17  sort     libc-2.23.so       [.] __random
     0.66%            16  sort     libc-2.23.so       [.] __random_r
     0.28%             7  sort     libc-2.23.so       [.] rand
     0.18%            17  sort     [kernel.kallsyms]  [.] native_irq_return_iret
```

```
     0.08%             2  sort     sort               [.] rand@plt
```

O Quick Sort pelo que se pode observar, passa o maior tempo a realizar o algoritmo em si e não a aceder a bibliotecas externas. O ini_vector e copy_vector e os acessos às bibliotecas de aleatoriedade são realizados em todos os sorts para a geração do vetor inicial.

Radix Sort

```
# Overhead       Samples  Command  Shared Object      Symbol
# ........  ............  .......  .................  ............................
#
    92.62%          1214  sort     sort               [.] sort2
     1.57%            27  sort     sort               [.] copy_vector
     1.31%            17  sort     libc-2.23.so       [.] __random
     1.05%            16  sort     libc-2.23.so       [.] __random_r
     1.01%            15  sort     sort               [.] ini_vector
     0.97%             5  sort     sort               [.] rand@plt
     0.74%            15  sort     [kernel.kallsyms]  [.] native_irq_return_iret
     0.73%            11  sort     libc-2.23.so       [.] rand
```

O Radix Sort, ao contrário do Quick Sort, passou mais consideravelmente mais tempo na função rand@plt, apesar de não ser utilizado qualquer tipo de randomização neste algoritmo. Por outro lado a percentagem de overhead aumentou pois o tempo que se passa a correr o algoritmo em si é menor no Radix Sort do que no Quick Sort.

Heap Sort

```
# Overhead       Samples  Command  Shared Object      Symbol
# ........  ............  .......  .................  ............................
#
    98.33%          5231  sort     sort               [.] sort3
     0.45%            20  sort     libc-2.23.so       [.] __random
     0.38%            28  sort     sort               [.] copy_vector
     0.37%            22  sort     sort               [.] ini_vector
     0.31%            20  sort     libc-2.23.so       [.] __random_r
     0.08%             5  sort     libc-2.23.so       [.] rand
     0.06%            14  sort     [kernel.kallsyms]  [.] native_irq_return_iret
     0.03%             2  sort     sort               [.] rand@plt
```

O Heap Sort, tal como o Quick Sort, passa quase todo o tempo a utilizar a função do algoritmo em si.

Merge Sort

```
# Overhead       Samples  Command  Shared Object      Symbol
# ........  ............  .......  .................  ............................
#
    86.59%          2976  sort     sort               [.] aux_sort4
```

```
3.10%          105  sort     sort               [.] sort4
2.11%           72  sort     libc-2.23.so       [.] _int_free
1.71%           59  sort     libc-2.23.so       [.] _int_malloc
1.52%           52  sort     libc-2.23.so       [.] malloc
0.96%           33  sort     libc-2.23.so       [.] free
0.77%           16  sort     libc-2.23.so       [.] __random_r
0.67%           23  sort     libc-2.23.so       [.] malloc_consolidate
0.66%           27  sort     sort               [.] copy_vector
0.62%           21  sort     libc-2.23.so       [.] __random
0.53%           27  sort     [kernel.kallsyms]  [.] native_irq_return_iret
0.44%           15  sort     sort               [.] ini_vector
0.20%            7  sort     libc-2.23.so       [.] rand
0.06%            2  sort     sort               [.] rand@plt
0.03%            1  sort     sort               [.] free@plt
0.03%            1  sort     sort               [.] malloc@plt
```

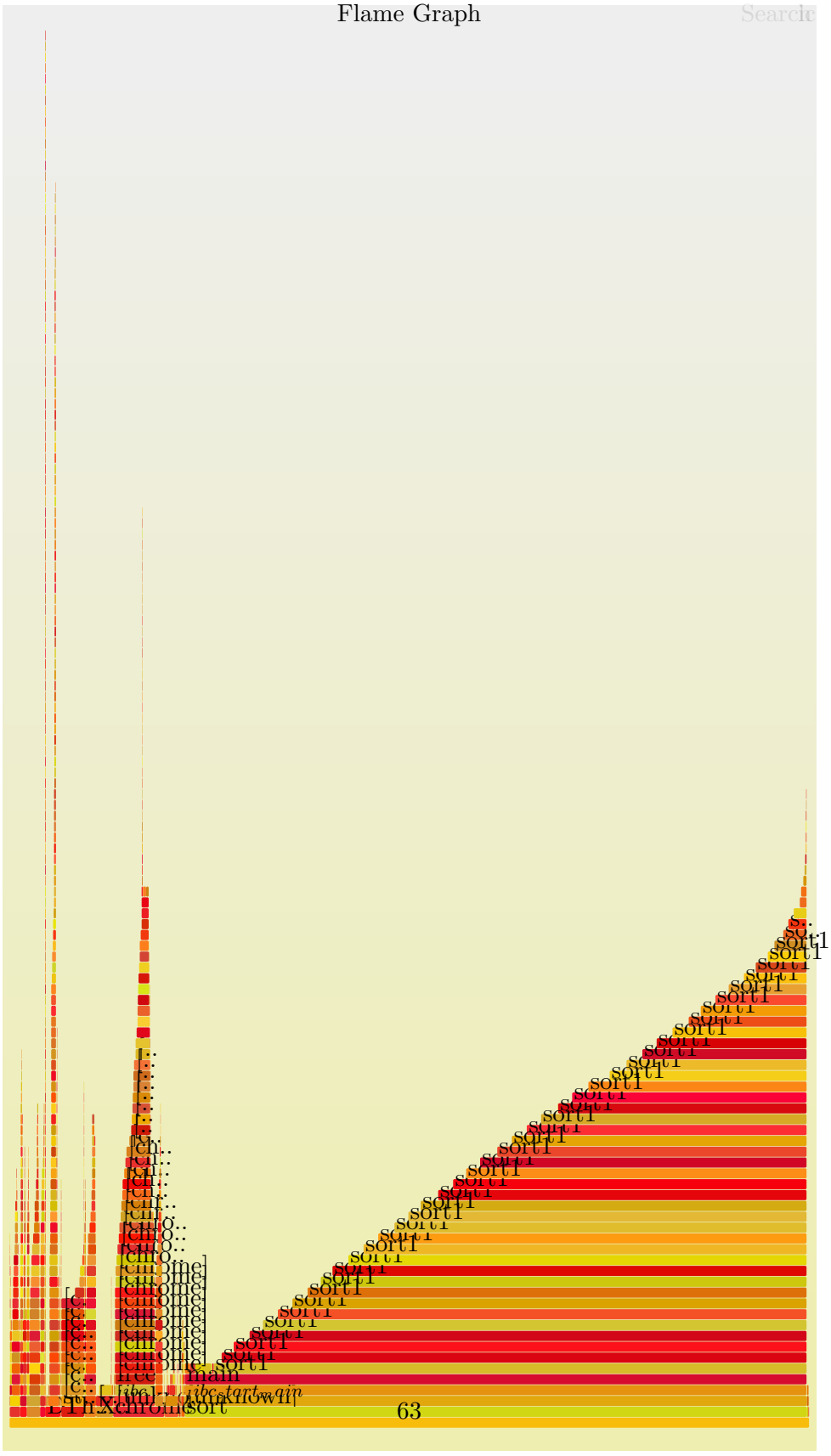
Devido ao facto de o Merge Sort possuir duas funções temos de ter em conta a soma do overhead das mesmas, para deduzir os acessos às bibliotecas. Mesmo somando o overhead das duas funções este não chega a tocar nos 90% o que quer dizer que este algoritmo recorre bastante a bibliotecas externas, isto devido ao facto de ele ter de alocar um array auxiliar em cada iteração, como tínhamos indicado anteriormente.

5.1.3 FlameGraphs

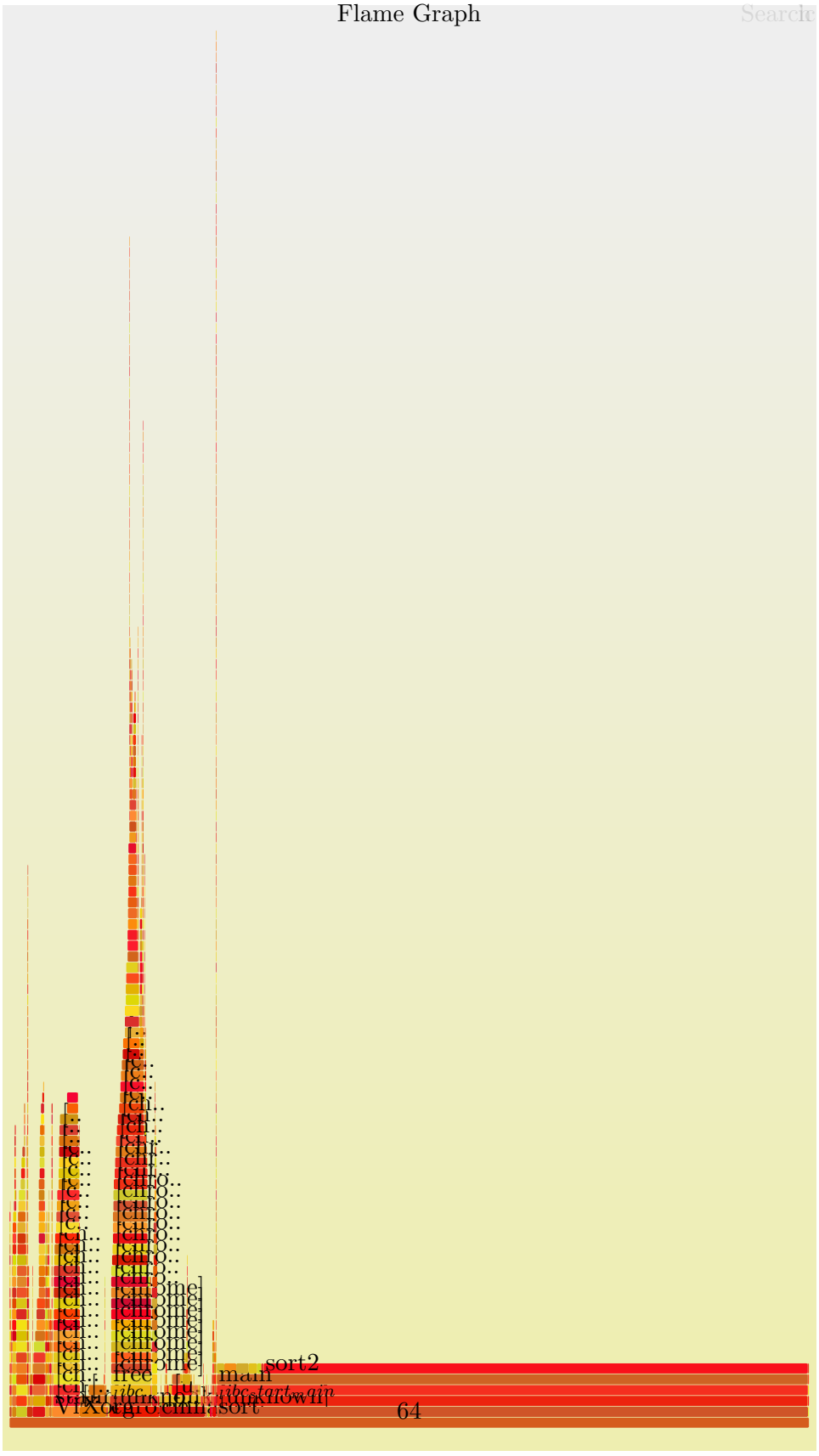
Para a obseração do comportamento de cada um destes algoritmos geraram-se flamegraphs para cada um deles.

Pode-se observar que tanto o RadixSort e HeapSort usam apenas a função de sorting em si, sendo assim difícil distinguí-los apartir do gráfico.

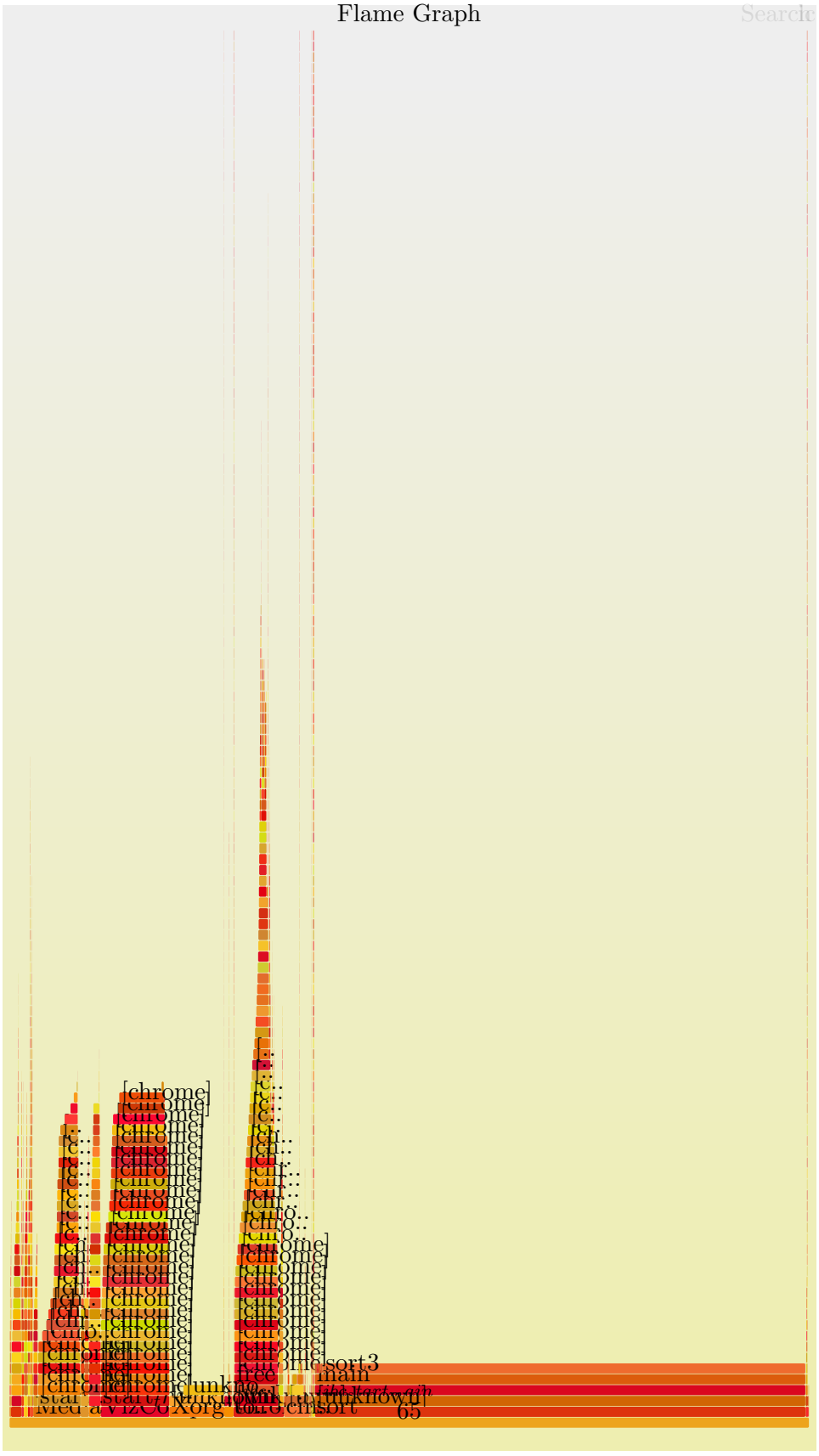
Pelo contrário, o QuickSort e o MergeSort resultam de recursão por isso é possível observar o todas as funções que foram chamadas para esse efeito. A grande diferença entre os dois é que o MergeSort além de chamar a função sort4 também chama a função auxiliar ao algoritmo o que garante assim a distinção entre os dois.



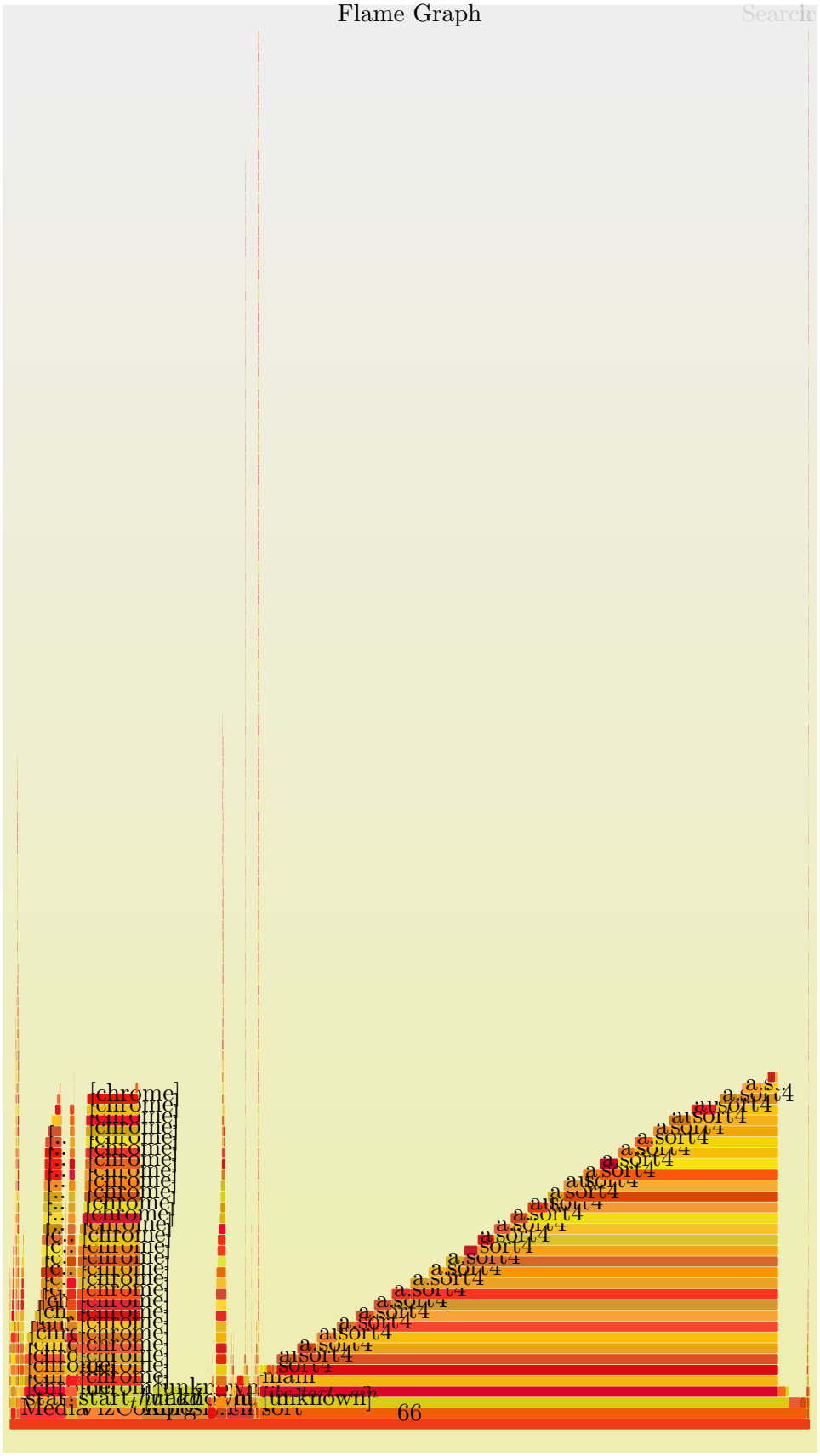
QuickSort FlameGraph



RadixSort FlameGraph



HeapSort FlameGraph



MergeSort FlameGraph

5.2 Parte 2 - Análise de Algoritmos de Multiplicação de Matrizes

Com ajuda dos tutoriais fornecidos pelo professor, foi-nos incumbida a tarefa de encontrar os pontos quentes de um programa de multiplicação de matrizes. Para tal tarefa definimos o tamanho de matriz como 2000x2000.

5.2.1 Encontrar os pontos quentes da execução

De forma a encontrar os pontos quentes da execução em primeiro lugar executou-se o comando perf stat de forma a obter alguns valores iniciais sobre o programa, tais como page-faults e tempo de execução.

```
> perf stat -e cpu-clock,faults ./naive

 Performance counter stats for './naive':

       5705.389200      cpu-clock (msec)
             1,083      faults

       5.712866600 seconds time elapsed
```

De seguida utilizou-se o comando perf record para obter o profile do programa em si, de forma a gerar reports mais tarde e obter o overhead das funções do programa.

```
> perf report --stdio --sort comm,dso

# Samples: 28K of event 'cpu-clock'
# Event count (approx.): 28939
#
# Overhead  Command      Shared Object
# ........  .......  ..................
#
    98.67%    naive  naive
     0.78%    naive  libc-2.12.so
     0.54%    naive  [kernel.kallsyms]
     0.00%    naive  ld-2.12.so


# Samples: 44  of event 'faults'
# Event count (approx.): 1086
#
# Overhead  Command      Shared Object
```

```
# ........  .......  ..................
#
    84.07%    naive  naive
    15.65%    naive  ld-2.12.so
     0.28%    naive  [kernel.kallsyms]
```

De forma a nos focarmos nos shared objects de interesse utilizou-se a flag -dsos de forma a limitar os Shared Objects de interesse.

```
> perf report --stdio --dsos=naive,libc-2.13.so

# Overhead  Command  Shared Object          Symbol
# ........  .......  .............  ..............
#
    98.62%    naive  naive          [.] main
     0.36%    naive  libc-2.12.so   [.] __random
     0.35%    naive  libc-2.12.so   [.] __random_r
     0.07%    naive  libc-2.12.so   [.] rand
     0.05%    naive  naive          [.] rand@plt


# Samples: 44  of event 'faults'
# Event count (approx.): 1086
#
# Overhead  Command  Shared Object    Symbol
# ........  .......  .............  ........
#
    84.07%    naive  naive          [.] main
```

Para encontrar os locais do programa onde o overhead é maior pode-se usar o comando perf annotate de forma a obter um perfil mais específico do programa em si.

```
> perf annotate --stdio --dsos=naive --symbol=main

 Percent | Source code & Disassembly of naive for cpu-clock
------------------------------------------------------------------
         :
         :
         :
         : Disassembly of section .text:
         :
         : 00000000004005f0 <main>:
```

```
      :     }
      :   }
      : }
      :
      : int main(int argc, char* argv[])
      : {
 0.00 :   4005f0:       push   %r14
 0.00 :   4005f2:       xor    %r14d,%r14d
 0.00 :   4005f5:       push   %r13
 0.00 :   4005f7:       push   %r12
 0.00 :   4005f9:       push   %rbp
 0.00 :   4005fa:       push   %rbx
      :       matrix_r[i][j] = sum ;
      :     }
      :   }
      : }
      :
      : int main(int argc, char* argv[])
 0.00 :   4005fb:       movslq %r14d,%rbp
 0.00 :   4005fe:       xor    %ebx,%ebx
 0.00 :   400600:       imul   $0x1f40,%rbp,%rbp
 0.00 :   400607:       lea    0x2485a20(%rbp),%r13
 0.00 :   40060e:       lea    0x1543620(%rbp),%r12
 0.00 :   400615:       add    $0x601220,%rbp
 0.00 :   40061c:       nopl   0x0(%rax)
      : {
      :   int i, j ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
 0.02 :   400620:       callq  4003c8 <rand@plt>
 0.02 :   400625:       cvtsi2ss %eax,%xmm0
      : void initialize_matrices()
      : {
      :   int i, j ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
 0.09 :   400629:       add    $0x1,%ebx
      :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
 0.00 :   40062c:       mulss  0x1c4(%rip),%xmm0        # 4007f8 <__dso_handle+0x
 0.06 :   400634:       movss  %xmm0,0x0(%r13)
      : void initialize_matrices()
```

```
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.03 :   40063a:       add    $0x4,%r13
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
     :       matrix_b[i][j] = (float) rand() / RAND_MAX ;
0.00 :   40063e:       callq  4003c8 <rand@plt>
0.04 :   400643:       cvtsi2ss %eax,%xmm0
     :       matrix_r[i][j] = 0.0 ;
0.07 :   400647:       movl   $0x0,0x0(%rbp)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.00 :   40064e:       add    $0x4,%rbp
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
     :       matrix_b[i][j] = (float) rand() / RAND_MAX ;
0.00 :   400652:       mulss  0x19e(%rip),%xmm0        # 4007f8 <__dso_handle+0x
0.10 :   40065a:       movss  %xmm0,(%r12)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.02 :   400660:       add    $0x4,%r12
0.00 :   400664:       cmp    $0x7d0,%ebx
0.00 :   40066a:       jne    400620 <main+0x30>
     :
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
0.00 :   40066c:       add    $0x1,%r14d
0.00 :   400670:       cmp    $0x7d0,%r14d
0.00 :   400677:       jne    4005fb <main+0xb>
     :     for (j = 0 ; j < MSIZE ; j++) {
     :       float sum = 0.0 ;
     :       for (k = 0 ; k < MSIZE ; k++) {
```

```
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
      :       }
      :       matrix_r[i][j] = sum ;
 0.00 :   400679:       xorps  %xmm2,%xmm2
      :
      : void initialize_matrices()
      : {
      :   int i, j ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
 0.00 :   40067c:       mov    $0x601220,%edi
 0.00 :   400681:       mov    $0x2485a20,%esi
      :
      : void multiply_matrices()
      : {
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
 0.00 :   400686:       xor    %ebx,%ebx
 0.00 :   400688:       nopl   0x0(%rax,%rax,1)
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
      :       }
      :       matrix_r[i][j] = sum ;
 0.00 :   400690:       xorps  %xmm1,%xmm1
 0.00 :   400693:       lea    0x1543620(%rbx),%rcx
 0.00 :   40069a:       mov    %rsi,%rdx
 0.00 :   40069d:       xor    %eax,%eax
 0.00 :   40069f:       nop
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
11.13 :   4006a0:       movaps %xmm2,%xmm0
 0.01 :   4006a3:       movlps (%rdx),%xmm0
 0.49 :   4006a6:       movhps 0x8(%rdx),%xmm0
 1.57 :   4006aa:       add    $0x4,%rdx
10.36 :   4006ae:       shufps $0x0,%xmm0,%xmm0
 0.29 :   4006b2:       mulps  (%rcx,%rax,1),%xmm0
56.69 :   4006b6:       add    $0x1f40,%rax
```

```
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
 7.00 :   4006bc:       cmp    $0xf42400,%rax
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
 0.01 :   4006c2:       addps  %xmm0,%xmm1
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
11.95 :   4006c5:       jne    4006a0 <main+0xb0>
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
      :       }
      :       matrix_r[i][j] = sum ;
 0.00 :   4006c7:       addps  %xmm2,%xmm1
 0.02 :   4006ca:       movaps %xmm1,(%rdi,%rbx,1)
 0.01 :   4006ce:       add    $0x10,%rbx
 0.00 :   4006d2:       cmp    $0x1f40,%rbx
 0.00 :   4006d9:       jne    400690 <main+0xa0>
 0.00 :   4006db:       add    $0x1f40,%rsi
 0.00 :   4006e2:       add    $0x1f40,%rdi
      :
      : void multiply_matrices()
      : {
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
 0.00 :   4006e9:       cmp    $0x33c7e20,%rsi
 0.00 :   4006f0:       jne    400686 <main+0x96>
      : int main(int argc, char* argv[])
      : {
      :   initialize_matrices() ;
      :   multiply_matrices() ;
      :   return( EXIT_SUCCESS ) ;
      : }
 0.00 :   4006f2:       pop    %rbx
 0.00 :   4006f3:       pop    %rbp
 0.00 :   4006f4:       pop    %r12
 0.00 :   4006f6:       pop    %r13
```

```
    0.00 :   4006f8:       xor    %eax,%eax
    0.00 :   4006fa:       pop    %r14
    0.00 :   4006fc:       retq
```

Com este comando pode-se observar que o overhead provém quase completamente da seguinte linha, a qual corresponde a aproximadamente 78% de todo o overhead do programa:

```
sum = sum + (matrix_a[i][k] * matrix_b[k][j]);
```

De seguida vamos aumentar a frequência de sampling, de forma a aumentar a precisão do profile.

```
> perf record -e cpu-clock --freq=8000 ./naive

> perf report --stdio --show-nr-samples --dsos=naive

# dso: naive
# Samples: 58K of event 'cpu-clock'
# Event count (approx.): 58075
#
# Overhead       Samples  Command        Symbol
# ........  ............  .......  ............
#
    98.55%         57232    naive  [.] main
     0.04%            24    naive  [.] rand@plt
```

Com todos estes resultados pode-se observar que muito provávelmente o slowdown provém de acessos lentos à memória.

5.2.2 Eventos de desempenho de hardware

Nesta segunda fase, tendo identificado o hot spot do programa em análise, decidiu-se realizar uma segunda implementação do mesmo, no qual a memória é acedida de maneira diferente. Particularmente alterou-se a ordem dos ciclos de i>j>k para i>k>j.

De seguida com a ajuda do comando perf stat obtiveram-se os seguintes valores:

```
Performance counter stats for './naive':

    20,482,364,019      instructions              #    1.07  insns per cycle
    19,085,647,497      cycles                    [62.47%]
     6,147,935,536      L1-dcache-loads
```

```
     2,131,206,220      L1-dcache-load-misses     #   34.67% of all L1-dcache hits
           99,460      dTLB-load-misses
       504,698,080      cache-references
         1,056,138      branch-misses             #    0.05% of all branches
     2,128,656,484      branch-instructions

       7.231122611 seconds time elapsed

Performance counter stats for './naive2':

    14,502,947,179      instructions              #    1.65  insns per cycle
     8,777,473,363      cycles                    [62.52%]
     4,145,861,213      L1-dcache-loads
       503,302,464      L1-dcache-load-misses     #   12.14% of all L1-dcache hits
            68,514      dTLB-load-misses
        78,895,754      cache-references
         4,036,699      branch-misses             #    0.19% of all branches
     2,129,233,291      branch-instructions

       3.336764310 seconds time elapsed
```

Apartir destes valores tornou-se possível a obtenção das seguintes métricas:

Métricas	naive	naive2
Instructions per cycle	1.07	1.65
L1 cache miss ratio	34.67%	12.14%
L1 cache miss rate PTI	104.05	34.7
Data TLB miss ratio	0.0001	0.0008
Data TLB miss rate PTI	0.005	0.005
Branch mispredict ratio	0.05%	0.19%
Branch mispredict rate PTI	0.05	0.27

Com estes valores pode-se observar que a segunda implementação possúi, além do óbvio melhoramento no tempo de execução, mais instruções por ciclo, e um número muito inferior de cache misses, sendo esta provávelmente a principal razão do slowdown.

5.2.3 Amostragem de eventos de desempenho de hardware

Esta fase consistia em aumentar a quantidade de samples de forma a não obter um overhead muito superior a 5%. Após alguns testes, cheguei a um número muito similar ao obtido no tutorial, 100000 samples. Apartir deste valor testei um perf report em cada um dos programas:

```
# =======
```

```
# captured on: Fri Jul  3 13:23:00 2020
# hostname : compute-431-8
# os release : 2.6.32-279.14.1.el6.x86_64
# perf version : 3.16.3-1.el6.elrepo.x86_64
# arch : x86_64
# nrcpus online : 24
# nrcpus avail : 24
# cpudesc : Intel(R) Xeon(R) CPU E5649 @ 2.53GHz
# cpuid : GenuineIntel,6,44,2
# total memory : 49551752 kB
# cmdline : /usr/bin/perf record -e cpu-cycles -c 100000 ./naive
# event : name = cpu-cycles, type = 0, config = 0x0, config1 = 0x0, config2 = 0x0, e
# HEADER_CPU_TOPOLOGY info available, use -I to display
# HEADER_NUMA_TOPOLOGY info available, use -I to display
# pmu mappings: cpu = 4, tracepoint = 2, software = 1
# ========
#
# Samples: 211K of event 'cpu-cycles'
# Event count (approx.): 21102000000
#
# Overhead       Samples  Command      Shared Object                              Symb
# ........  ............  .......  .................  ..............................
#
    98.76%        208393    naive  naive              [.] main
     0.36%           767    naive  libc-2.12.so       [.] __random
     0.31%           646    naive  libc-2.12.so       [.] __random_r
     0.10%           212    naive  [kernel.kallsyms]  [k] clear_page_c


# ========
# captured on: Fri Jul  3 13:30:17 2020
# hostname : compute-431-8
# os release : 2.6.32-279.14.1.el6.x86_64
# perf version : 3.16.3-1.el6.elrepo.x86_64
# arch : x86_64
# nrcpus online : 24
# nrcpus avail : 24
# cpudesc : Intel(R) Xeon(R) CPU E5649 @ 2.53GHz
# cpuid : GenuineIntel,6,44,2
# total memory : 49551752 kB
# cmdline : /usr/bin/perf record -e cpu-cycles -c 100000 ./naive2
# event : name = cpu-cycles, type = 0, config = 0x0, config1 = 0x0, config2 = 0x0, e
# HEADER_CPU_TOPOLOGY info available, use -I to display
# HEADER_NUMA_TOPOLOGY info available, use -I to display
```

```
# pmu mappings: cpu = 4, tracepoint = 2, software = 1
# ========
#
# Samples: 108K of event 'cpu-cycles'
# Event count (approx.): 10889700000
#
# Overhead       Samples  Command      Shared Object                              S
# ........  ............  .......  .................  ..............................
#
    97.62%        106308   naive2  naive2             [.] main
     0.72%           781   naive2  libc-2.12.so       [.] __random
     0.60%           656   naive2  libc-2.12.so       [.] __random_r
     0.19%           208   naive2  [kernel.kallsyms]  [k] clear_page_c
     0.14%           152   naive2  [kernel.kallsyms]  [k] hrtimer_interrupt
     0.12%           136   naive2  libc-2.12.so       [.] rand
```

Após este feito procedi a analisar de novo o overhead no código com perf annotate:

```
 Percent | Source code & Disassembly of naive for cpu-cycles
------------------------------------------------------------------
         :
         :
         :
         : Disassembly of section .text:
         :
         : 00000000004005f0 <main>:
         :     }
         :   }
         : }
         :
         : int main(int argc, char* argv[])
         : {
    0.00 :   4005f0:       push   %r14
    0.00 :   4005f2:       xor    %r14d,%r14d
    0.00 :   4005f5:       push   %r13
    0.00 :   4005f7:       push   %r12
    0.00 :   4005f9:       push   %rbp
    0.00 :   4005fa:       push   %rbx
         :       matrix_r[i][j] = sum ;
         :     }
         :   }
         : }
         :
```

```
     : int main(int argc, char* argv[])
0.00 :   4005fb:       movslq %r14d,%rbp
0.00 :   4005fe:       xor    %ebx,%ebx
0.00 :   400600:       imul   $0x1f40,%rbp,%rbp
0.00 :   400607:       lea    0x2485a20(%rbp),%r13
0.00 :   40060e:       lea    0x1543620(%rbp),%r12
0.00 :   400615:       add    $0x601220,%rbp
0.00 :   40061c:       nopl   0x0(%rax)
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
0.01 :   400620:       callq  4003c8 <rand@plt>
0.02 :   400625:       cvtsi2ss %eax,%xmm0
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.04 :   400629:       add    $0x1,%ebx
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
0.00 :   40062c:       mulss  0x1c4(%rip),%xmm0        # 4007f8 <__dso_handle+0x
0.07 :   400634:       movss  %xmm0,0x0(%r13)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.01 :   40063a:       add    $0x4,%r13
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
     :       matrix_b[i][j] = (float) rand() / RAND_MAX ;
0.00 :   40063e:       callq  4003c8 <rand@plt>
0.02 :   400643:       cvtsi2ss %eax,%xmm0
     :       matrix_r[i][j] = 0.0 ;
0.04 :   400647:       movl   $0x0,0x0(%rbp)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
```

```
     :     for (j = 0 ; j < MSIZE ; j++) {
0.00 :   40064e:       add    $0x4,%rbp
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
     :       matrix_b[i][j] = (float) rand() / RAND_MAX ;
0.00 :   400652:       mulss  0x19e(%rip),%xmm0        # 4007f8 <__dso_handle+0x
0.06 :   40065a:       movss  %xmm0,(%r12)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.02 :   400660:       add    $0x4,%r12
0.00 :   400664:       cmp    $0x7d0,%ebx
0.00 :   40066a:       jne    400620 <main+0x30>
     :
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
0.00 :   40066c:       add    $0x1,%r14d
0.00 :   400670:       cmp    $0x7d0,%r14d
0.00 :   400677:       jne    4005fb <main+0xb>
     :     for (j = 0 ; j < MSIZE ; j++) {
     :       float sum = 0.0 ;
     :       for (k = 0 ; k < MSIZE ; k++) {
     :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
     :       }
     :       matrix_r[i][j] = sum ;
0.00 :   400679:       xorps  %xmm2,%xmm2
     :
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
0.00 :   40067c:       mov    $0x601220,%edi
0.00 :   400681:       mov    $0x2485a20,%esi
     :
     : void multiply_matrices()
     : {
     :   int i, j, k ;
     :
```

```
      :   for (i = 0 ; i < MSIZE ; i++) {
 0.00 :   400686:       xor    %ebx,%ebx
 0.00 :   400688:       nopl   0x0(%rax,%rax,1)
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
      :       }
      :       matrix_r[i][j] = sum ;
 0.01 :   400690:       xorps  %xmm1,%xmm1
 0.00 :   400693:       lea    0x1543620(%rbx),%rcx
 0.00 :   40069a:       mov    %rsi,%rdx
 0.00 :   40069d:       xor    %eax,%eax
 0.01 :   40069f:       nop
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
 9.73 :   4006a0:       movaps %xmm2,%xmm0
 0.00 :   4006a3:       movlps (%rdx),%xmm0
 0.46 :   4006a6:       movhps 0x8(%rdx),%xmm0
 1.52 :   4006aa:       add    $0x4,%rdx
 9.44 :   4006ae:       shufps $0x0,%xmm0,%xmm0
 0.27 :   4006b2:       mulps  (%rcx,%rax,1),%xmm0
60.16 :   4006b6:       add    $0x1f40,%rax
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
 6.19 :   4006bc:       cmp    $0xf42400,%rax
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
 0.00 :   4006c2:       addps  %xmm0,%xmm1
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (j = 0 ; j < MSIZE ; j++) {
      :       float sum = 0.0 ;
      :       for (k = 0 ; k < MSIZE ; k++) {
11.90 :   4006c5:       jne    4006a0 <main+0xb0>
      :         sum = sum + (matrix_a[i][k] * matrix_b[k][j]) ;
```

```
      :       }
      :       matrix_r[i][j] = sum ;
 0.00 :   4006c7:       addps  %xmm2,%xmm1
 0.02 :   4006ca:       movaps %xmm1,(%rdi,%rbx,1)
 0.00 :   4006ce:       add    $0x10,%rbx
 0.00 :   4006d2:       cmp    $0x1f40,%rbx
 0.00 :   4006d9:       jne    400690 <main+0xa0>
 0.00 :   4006db:       add    $0x1f40,%rsi
 0.00 :   4006e2:       add    $0x1f40,%rdi
      :
      : void multiply_matrices()
      : {
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
 0.00 :   4006e9:       cmp    $0x33c7e20,%rsi
 0.00 :   4006f0:       jne    400686 <main+0x96>
      : int main(int argc, char* argv[])
      : {
      :   initialize_matrices() ;
      :   multiply_matrices() ;
      :   return( EXIT_SUCCESS ) ;
      : }
 0.00 :   4006f2:       pop    %rbx
 0.00 :   4006f3:       pop    %rbp
 0.00 :   4006f4:       pop    %r12
 0.00 :   4006f6:       pop    %r13
 0.00 :   4006f8:       xor    %eax,%eax
 0.00 :   4006fa:       pop    %r14
 0.00 :   4006fc:       retq

 Percent | Source code & Disassembly of naive2 for cpu-cycles
-----------------------------------------------------------------------
      :
      :
      :
      : Disassembly of section .text:
      :
      : 00000000004005f0 <main>:
      :     }
      :   }
      : }
      :
      : int main(int argc, char* argv[])
```

```
     : {
0.00 :   4005f0:       push   %r14
0.00 :   4005f2:       xor    %r14d,%r14d
0.00 :   4005f5:       push   %r13
0.00 :   4005f7:       push   %r12
0.00 :   4005f9:       push   %rbp
0.00 :   4005fa:       push   %rbx
     :
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
0.00 :   4005fb:       movslq %r14d,%rbp
0.00 :   4005fe:       xor    %ebx,%ebx
0.00 :   400600:       imul   $0x1f40,%rbp,%rbp
0.00 :   400607:       lea    0x2485a20(%rbp),%r13
0.00 :   40060e:       lea    0x1543620(%rbp),%r12
0.00 :   400615:       add    $0x601220,%rbp
0.00 :   40061c:       nopl   0x0(%rax)
     :     for (j = 0 ; j < MSIZE ; j++) {
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
0.03 :   400620:       callq  4003c8 <rand@plt>
0.04 :   400625:       cvtsi2ss %eax,%xmm0
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.09 :   400629:       add    $0x1,%ebx
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
0.00 :   40062c:       mulss  0x1c4(%rip),%xmm0        # 4007f8 <__dso_handle+0x
0.12 :   400634:       movss  %xmm0,0x0(%r13)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.04 :   40063a:       add    $0x4,%r13
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
     :       matrix_b[i][j] = (float) rand() / RAND_MAX ;
0.00 :   40063e:       callq  4003c8 <rand@plt>
```

```
0.03 :   400643:       cvtsi2ss %eax,%xmm0
     :       matrix_r[i][j] = 0.0 ;
0.10 :   400647:       movl   $0x0,0x0(%rbp)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.00 :   40064e:       add    $0x4,%rbp
     :       matrix_a[i][j] = (float) rand() / RAND_MAX ;
     :       matrix_b[i][j] = (float) rand() / RAND_MAX ;
0.00 :   400652:       mulss  0x19e(%rip),%xmm0        # 4007f8 <__dso_handle+0x
0.12 :   40065a:       movss  %xmm0,(%r12)
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
     :     for (j = 0 ; j < MSIZE ; j++) {
0.03 :   400660:       add    $0x4,%r12
0.00 :   400664:       cmp    $0x7d0,%ebx
0.00 :   40066a:       jne    400620 <main+0x30>
     :
     : void initialize_matrices()
     : {
     :   int i, j ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
0.00 :   40066c:       add    $0x1,%r14d
0.00 :   400670:       cmp    $0x7d0,%r14d
0.00 :   400677:       jne    4005fb <main+0xb>
0.00 :   400679:       mov    $0x601220,%edx
0.00 :   40067e:       xor    %esi,%esi
     :
     : void multiply_matrices()
     : {
     :   int i, j, k ;
     :
     :   for (i = 0 ; i < MSIZE ; i++) {
0.00 :   400680:       movslq %esi,%rbx
0.00 :   400683:       mov    $0x1543620,%ecx
0.00 :   400688:       imul   $0x1f40,%rbx,%rbx
0.00 :   40068f:       add    $0x2485a20,%rbx
```

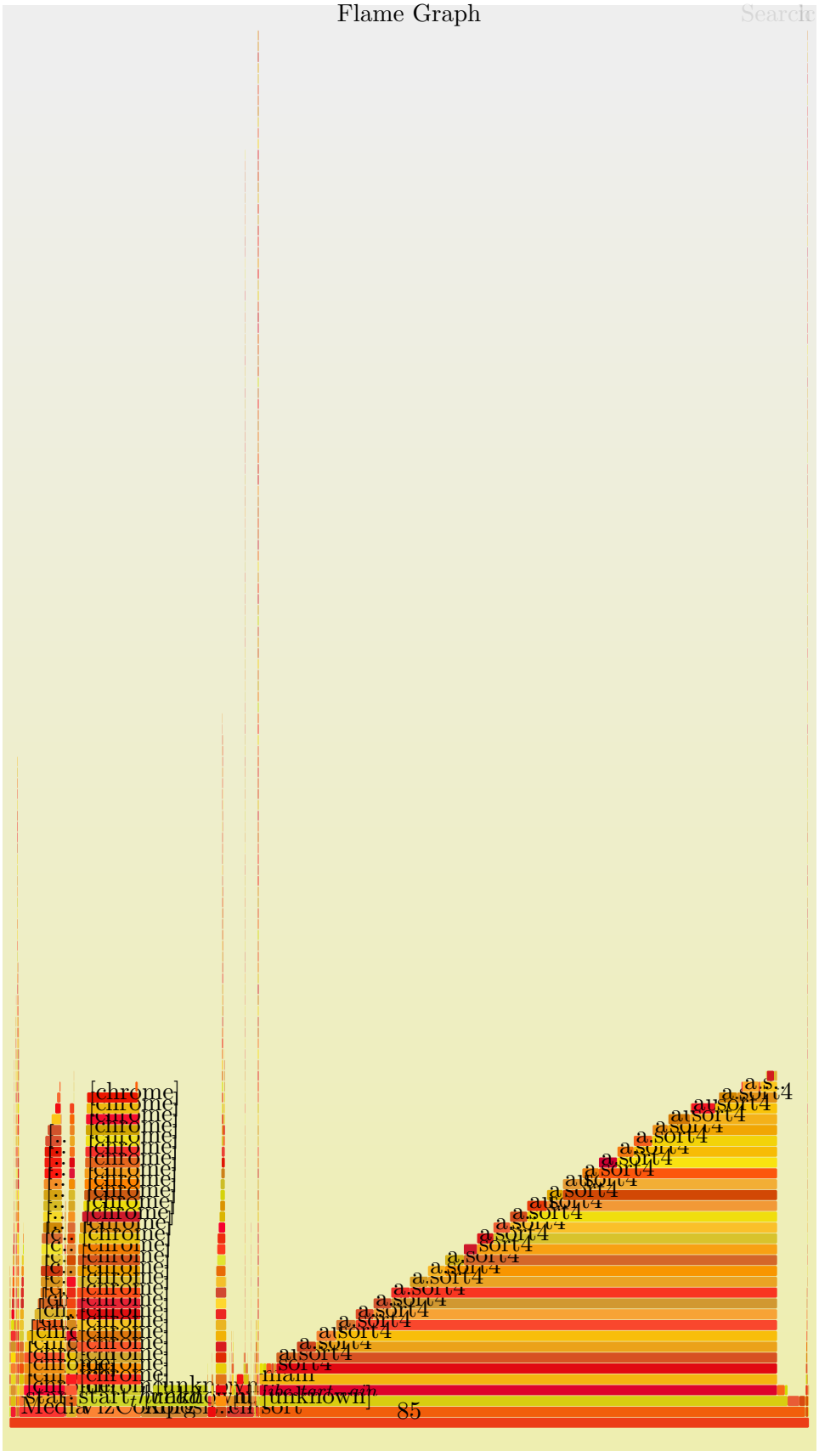
```
 0.00 :   400696:       nopw   %cs:0x0(%rax,%rax,1)
      :     for (k = 0 ; k < MSIZE ; k++) {
 0.04 :   4006a0:       movss  (%rbx),%xmm1
 0.02 :   4006a4:       xor    %eax,%eax
 0.00 :   4006a6:       shufps $0x0,%xmm1,%xmm1
 0.02 :   4006aa:       nopw   0x0(%rax,%rax,1)
      :       for (j = 0 ; j < MSIZE ; j++) {
      :         matrix_r[i][j] = matrix_r[i][j] + (matrix_a[i][k] * matrix_b[k][j
20.55 :   4006b0:       movaps (%rcx,%rax,1),%xmm0
35.26 :   4006b4:       mulps  %xmm1,%xmm0
10.85 :   4006b7:       addps  (%rdx,%rax,1),%xmm0
27.71 :   4006bb:       movaps %xmm0,(%rdx,%rax,1)
 4.88 :   4006bf:       add    $0x10,%rax
 0.02 :   4006c3:       cmp    $0x1f40,%rax
 0.02 :   4006c9:       jne    4006b0 <main+0xc0>
 0.04 :   4006cb:       add    $0x1f40,%rcx
 0.01 :   4006d2:       add    $0x4,%rbx
      : void multiply_matrices()
      : {
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
      :     for (k = 0 ; k < MSIZE ; k++) {
 0.00 :   4006d6:       cmp    $0x2485a20,%rcx
 0.00 :   4006dd:       jne    4006a0 <main+0xb0>
      :
      : void multiply_matrices()
      : {
      :   int i, j, k ;
      :
      :   for (i = 0 ; i < MSIZE ; i++) {
 0.00 :   4006df:       add    $0x1,%esi
 0.00 :   4006e2:       add    $0x1f40,%rdx
 0.00 :   4006e9:       cmp    $0x7d0,%esi
 0.00 :   4006ef:       jne    400680 <main+0x90>
      : int main(int argc, char* argv[])
      : {
      :   initialize_matrices() ;
      :   multiply_matrices() ;
      :   return( EXIT_SUCCESS ) ;
      : }
 0.00 :   4006f1:       pop    %rbx
 0.00 :   4006f2:       pop    %rbp
 0.00 :   4006f3:       pop    %r12
```

```
0.00 :   4006f5:       pop    %r13
0.00 :   4006f7:       xor    %eax,%eax
0.00 :   4006f9:       pop    %r14
0.00 :   4006fb:       retq
```

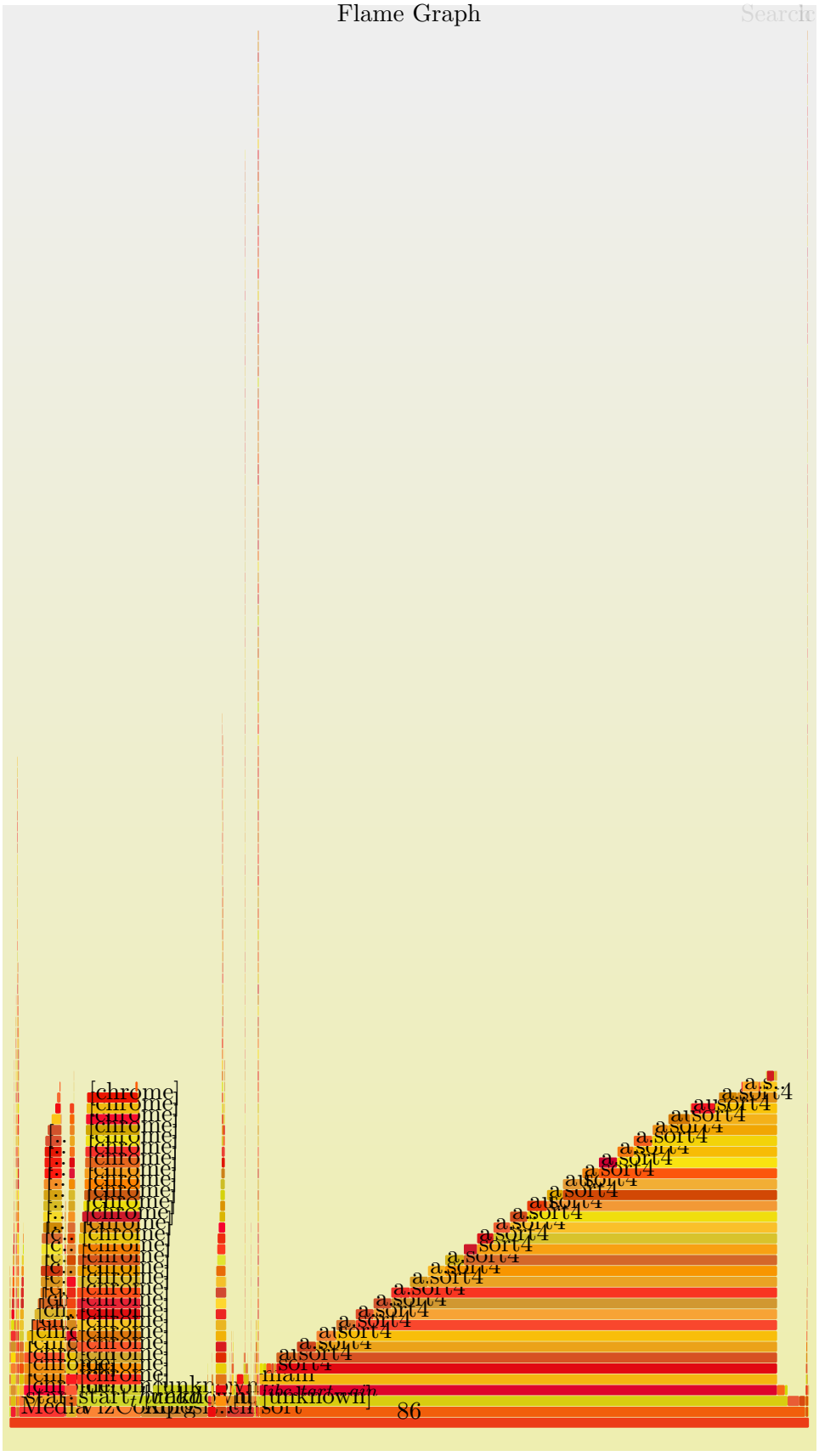
Aqui observa-se com garantia de uma melhor avaliação o overhead recebido por cada linha de código, devido a um valor de sampling superior.

5.2.4 Geração de FlameGraphs

Com a ajuda da ferramenta FlameGraphs geraram-se os seguintes gráficos de desempenho de cada um dos programas:



Naive


Flame Graph
Search
[chrome]
a.sort4
sort4
main
[unknown]
__libc_start_main
start
86

Naive 2

Capítulo 6

Conclusão

Com o uso destas ferramentas apercebemo-nos da potência das mesmas e do quão úteis estas podem ser na optimização e análise de programas.

A ferramenta DTrace permite ao utilizador uma grande costumização de análise, permitindo obter valores não só prédefinidos como também resultados de pontas de proba definidas pelo utilizador que podem retornar resultados interiores ao programa e devolver uma análise muito mais personalizada do que as outras ferramentas.

Em relação à ferramenta perf, esta é uma poderosa ferramenta que permite ao utilizador uma análise a fundo do programa chegando até ao overhead de cada linha de código em assembly, garantindo assim uma análise bastante exata dos hotspots presentes em cada programa.

Desta maneira pode-se observar que todas estas ferramentas podem ser utilizadas para fins diferentes e que são uma grande adição à caixa de ferramentas de qualquer programador preocupado com o desempenho das suas aplicações.

Appendices

Apêndice A

Anexos

vmstat

data.txt

```
    66068588  total memory
     4215292  used memory
     1665188  active memory
     1251372  inactive memory
    61853296  free memory
      235908  buffer memory
     2487940  swap cache
     1023992  total swap
       13728  used swap
     1010264  free swap
  5923119558 non-nice user cpu ticks
       23926 nice user cpu ticks
   162530633 system cpu ticks
 42114169051 idle cpu ticks
    10312497 IO-wait cpu ticks
          20 IRQ cpu ticks
     1818638 softirq cpu ticks
           0 stolen cpu ticks
     1496869 pages paged in
   148004300 pages paged out
       66509 pages swapped in
     3446049 pages swapped out
  3851325965 interrupts
  2248152821 CPU context switches
  1577447408 boot time
    12262554 forks
```

top

data.txt

```
21473 a77230    20   0  209m 190m  904 R 97.0  0.3   0:01.53 sp-mz.B.x
21482 a77230    20   0 15816 1880  820 R  3.8  0.0   0:00.03 top
21174 a77230    20   0 11328 1392 1160 S  0.0  0.0   0:00.00 bash
21238 a77230    20   0  9228 1220 1020 S  0.0  0.0   0:00.00 717050.search6.
21480 a77230    20   0  9228  444  240 S  0.0  0.0   0:00.00 717050.search6.
21483 a77230    20   0  6376  664  580 S  0.0  0.0   0:00.00 grep
```

ps

data.txt

```
  PID TTY          TIME CMD
21174 ?        00:00:00 bash
21238 ?        00:00:00 717050.search6.
21472 ?        00:00:00 717050.search6.
21473 ?        00:00:01 sp-mz.B.x
21475 ?        00:00:00 717050.search6.
21479 ?        00:00:00 ps
21480 ?        00:00:00 717050.search6.
21481 ?        00:00:00 lsof
21482 ?        00:00:00 top
21483 ?        00:00:00 grep
```

mpstat

data.txt

```
Linux 2.6.32-279.14.1.el6.x86_64 compute-662-6.local         04/21/20         _x86_64_        48 CPU

21:14:38     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
21:14:38     all   12.29    0.00    0.34    0.02    0.00    0.00    0.00    0.00   87.35
```

iostat

data.txt

```
Linux 2.6.32-279.14.1.el6.x86_64 compute-662-6.local         04/21/20         _x86_64_        48 CPU

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          12.29    0.00    0.34    0.02    0.00   87.35

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.83         0.30        29.45    2991306  296016884
```


lsof

data.txt

```
COMMAND     PID   USER   FD   TYPE DEVICE SIZE/OFF     NODE NAME
bash      21174 a77230  cwd    DIR   0,21     4096  9737154 /home/a77230 nas-0-0.local:/state/partition1/cpd-2019/a77230
bash      21174 a77230  rtd    DIR    8,1     4096        2 /
bash      21174 a77230  txt    REG    8,1   938736   524432 /bin/bash
bash      21174 a77230  mem    REG    8,1   156872   281686 /lib64/ld-2.12.so
bash      21174 a77230  mem    REG    8,1  1922112   281687 /lib64/libc-2.12.so
bash      21174 a77230  mem    REG    8,1    22536   281689 /lib64/libdl-2.12.so
bash      21174 a77230  mem    REG    8,1   138280   262263 /lib64/libtinfo.so.5.7
bash      21174 a77230  mem    REG    8,1    65928   262179 /lib64/libnss_files-2.12.so
bash      21174 a77230    0r  FIFO    0,8      0t0 78553490 pipe
bash      21174 a77230    1w   REG    8,2     2740   128996 /var/spool/torque/spool/717050.search6.di.uminho.pt.OU
bash      21174 a77230    2w   REG    8,2        0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
```

```
bash      21174 a77230   18w  FIFO    0,8        0t0 78553492 pipe
717050.se 21238 a77230   cwd   DIR   0,21       4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
717050.se 21238 a77230   rtd   DIR    8,1       4096        2 /
717050.se 21238 a77230   txt   REG    8,1     938736   524432 /bin/bash
717050.se 21238 a77230   mem   REG    8,1     156872   281686 /lib64/ld-2.12.so
717050.se 21238 a77230   mem   REG    8,1    1922112   281687 /lib64/libc-2.12.so
717050.se 21238 a77230   mem   REG    8,1      22536   281689 /lib64/libdl-2.12.so
717050.se 21238 a77230   mem   REG    8,1     138280   262263 /lib64/libtinfo.so.5.7
717050.se 21238 a77230    0r  FIFO    0,8        0t0 78553490 pipe
717050.se 21238 a77230    1w   REG    8,2       2740   128996 /var/spool/torque/spool/717050.search6.di.uminho.pt.OU
717050.se 21238 a77230    2w   REG    8,2          0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
717050.se 21238 a77230   18w  FIFO    0,8        0t0 78553492 pipe
717050.se 21238 a77230   255r  REG    8,2       2010   128785 /var/spool/torque/mom_priv/jobs/717050.search6.di.uminho.pt.SC
717050.se 21472 a77230   cwd   DIR   0,21       4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
717050.se 21472 a77230   rtd   DIR    8,1       4096        2 /
717050.se 21472 a77230   txt   REG    8,1     938736   524432 /bin/bash
717050.se 21472 a77230   mem   REG    8,1     156872   281686 /lib64/ld-2.12.so
717050.se 21472 a77230   mem   REG    8,1    1922112   281687 /lib64/libc-2.12.so
717050.se 21472 a77230   mem   REG    8,1      22536   281689 /lib64/libdl-2.12.so
717050.se 21472 a77230   mem   REG    8,1     138280   262263 /lib64/libtinfo.so.5.7
717050.se 21472 a77230    0r  CHR    1,3        0t0     3925 /dev/null
717050.se 21472 a77230    1w   REG    8,2       2740   128996 /var/spool/torque/spool/717050.search6.di.uminho.pt.OU
717050.se 21472 a77230    2w   REG    8,2          0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
717050.se 21472 a77230   18w  FIFO    0,8        0t0 78553492 pipe
sp-mz.B.x 21473 a77230   cwd   DIR   0,21       4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
sp-mz.B.x 21473 a77230   rtd   DIR    8,1       4096        2 /
sp-mz.B.x 21473 a77230   txt   REG   0,21     107047  9738498 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER/bin/sp-mz.B.x nas-0-0.local:/state/partition1/cpd-
sp-mz.B.x 21473 a77230   mem   REG    8,1     156872   281686 /lib64/ld-2.12.so
sp-mz.B.x 21473 a77230   mem   REG    8,1    1922112   281687 /lib64/libc-2.12.so
sp-mz.B.x 21473 a77230   mem   REG    8,1     598800   281692 /lib64/libm-2.12.so
sp-mz.B.x 21473 a77230   mem   REG   0,22    5450945 54136780 /share/apps/gcc/5.3.0/lib64/libgfortran.so.3.0.0 nas-1-1:/storage/local/d2/v0p1/
sp-mz.B.x 21473 a77230   mem   REG   0,22     529980 54136777 /share/apps/gcc/5.3.0/lib64/libgcc_s.so.1 nas-1-1:/storage/local/d2/v0p1/
```

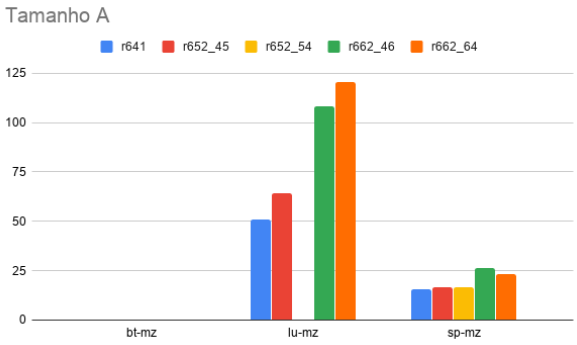
```
sp-mz.B.x 21473 a77230  mem    REG   0,22   931674 54136797 /share/apps/gcc/5.3.0/lib64/libquadmath.so.0.0.0 nas-1-1:/storage/local/d2/v0p1/
sp-mz.B.x 21473 a77230    0r  FIFO    0,8      0t0 78553490 pipe
sp-mz.B.x 21473 a77230    1w   REG   0,21        0  9738700 /home/a77230/ESC/NPB3.3.1-MZ/RESULTS_FINAL/NPB3.3-MZ-SER/sp-mz.B/output.txt nas-0-0.local:/st
sp-mz.B.x 21473 a77230    2w   REG    8,2        0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
sp-mz.B.x 21473 a77230   18w  FIFO    0,8      0t0 78553492 pipe
717050.se 21475 a77230  cwd    DIR   0,21     4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
717050.se 21475 a77230  rtd    DIR    8,1     4096        2 /
717050.se 21475 a77230  txt    REG    8,1   938736   524432 /bin/bash
717050.se 21475 a77230  mem    REG    8,1   156872   281686 /lib64/ld-2.12.so
717050.se 21475 a77230  mem    REG    8,1  1922112   281687 /lib64/libc-2.12.so
717050.se 21475 a77230  mem    REG    8,1    22536   281689 /lib64/libdl-2.12.so
717050.se 21475 a77230  mem    REG    8,1   138280   262263 /lib64/libtinfo.so.5.7
717050.se 21475 a77230    0r   CHR    1,3      0t0     3925 /dev/null
717050.se 21475 a77230    1w   REG    8,2     2740   128996 /var/spool/torque/spool/717050.search6.di.uminho.pt.OU
717050.se 21475 a77230    2w   REG    8,2        0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
717050.se 21475 a77230   18w  FIFO    0,8      0t0 78553492 pipe
717050.se 21480 a77230  cwd    DIR   0,21     4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
717050.se 21480 a77230  rtd    DIR    8,1     4096        2 /
717050.se 21480 a77230  txt    REG    8,1   938736   524432 /bin/bash
717050.se 21480 a77230  mem    REG    8,1   156872   281686 /lib64/ld-2.12.so
717050.se 21480 a77230  mem    REG    8,1  1922112   281687 /lib64/libc-2.12.so
717050.se 21480 a77230  mem    REG    8,1    22536   281689 /lib64/libdl-2.12.so
717050.se 21480 a77230  mem    REG    8,1   138280   262263 /lib64/libtinfo.so.5.7
717050.se 21480 a77230    0r   CHR    1,3      0t0     3925 /dev/null
717050.se 21480 a77230    1w   REG    8,2     2740   128996 /var/spool/torque/spool/717050.search6.di.uminho.pt.OU
717050.se 21480 a77230    2w   REG    8,2        0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
717050.se 21480 a77230   18w  FIFO    0,8      0t0 78553492 pipe
lsof      21481 a77230  cwd    DIR   0,21     4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
lsof      21481 a77230  rtd    DIR    8,1     4096        2 /
lsof      21481 a77230  txt    REG    8,1   145872   802498 /usr/sbin/lsof
lsof      21481 a77230  mem    REG    8,1   156872   281686 /lib64/ld-2.12.so
lsof      21481 a77230  mem    REG    8,1  1922112   281687 /lib64/libc-2.12.so
```

```
lsof  21481 a77230  mem    REG    8,1    22536  281689 /lib64/libdl-2.12.so
lsof  21481 a77230  mem    REG    8,1   124624  281706 /lib64/libselinux.so.1
lsof  21481 a77230  mem    REG    8,1    65928  262179 /lib64/libnss_files-2.12.so
lsof  21481 a77230    0r   CHR    1,3      0t0    3925 /dev/null
lsof  21481 a77230    1w   REG   0,21        0 9738708 /home/a77230/ESC/NPB3.3.1-MZ/RESULTS_FINAL/NPB3.3-MZ-SER/sp-mz.B/lsof.txt nas-0-0.local:/stat
lsof  21481 a77230    2w   REG    8,2        0  128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
lsof  21481 a77230    3r   DIR    0,3        0       1 /proc
lsof  21481 a77230    4r   DIR    0,3        0 78554017 /proc/21481/fd
lsof  21481 a77230    5w  FIFO    0,8      0t0 78554022 pipe
lsof  21481 a77230    6r  FIFO    0,8      0t0 78554023 pipe
top   21482 a77230  cwd    DIR   0,21     4096 9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
top   21482 a77230  rtd    DIR    8,1     4096       2 /
top   21482 a77230  txt    REG    8,1    68392  795304 /usr/bin/top
top   21482 a77230  mem    REG    8,1   156872  281686 /lib64/ld-2.12.so
top   21482 a77230  mem    REG    8,1  1922112  281687 /lib64/libc-2.12.so
top   21482 a77230  mem    REG    8,1    22536  281689 /lib64/libdl-2.12.so
top   21482 a77230  mem    REG    8,1    65608  269785 /lib64/libproc-3.2.8.so
top   21482 a77230  mem    REG    8,1   195616  262285 /lib64/libncursesw.so.5.7
top   21482 a77230  mem    REG    8,1   138280  262263 /lib64/libtinfo.so.5.7
top   21482 a77230  mem    REG    8,1    65928  262179 /lib64/libnss_files-2.12.so
top   21482 a77230    0r   CHR    1,3      0t0    3925 /dev/null
top   21482 a77230    1w  FIFO    0,8      0t0 78553999 pipe
top   21482 a77230    2w   REG    8,2        0  128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
top   21482 a77230   18w  FIFO    0,8      0t0 78553492 pipe
grep  21483 a77230  cwd    DIR   0,21     4096 9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
grep  21483 a77230  rtd    DIR    8,1     4096       2 /
grep  21483 a77230  txt    REG    8,1   111360  544212 /bin/grep
grep  21483 a77230  mem    REG    8,1   156872  281686 /lib64/ld-2.12.so
grep  21483 a77230  mem    REG    8,1  1922112  281687 /lib64/libc-2.12.so
grep  21483 a77230  mem    REG    8,1   183848  262201 /lib64/libpcre.so.0.0.1
grep  21483 a77230    0r  FIFO    0,8      0t0 78553999 pipe
grep  21483 a77230    1w   REG   0,21        0 9738711 /home/a77230/ESC/NPB3.3.1-MZ/RESULTS_FINAL/NPB3.3-MZ-SER/sp-mz.B/top.txt nas-0-0.local:/state
```

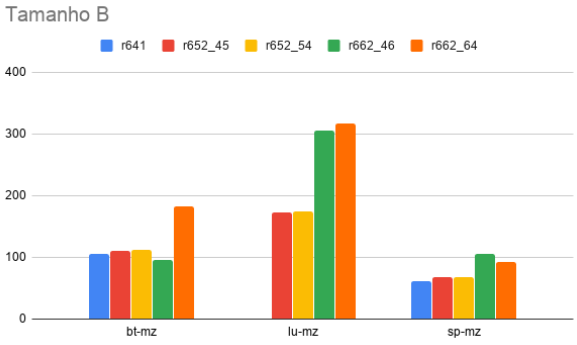
```
grep      21483 a77230    2w   REG    8,2        0   128999 /var/spool/torque/spool/717050.search6.di.uminho.pt.ER
grep      21483 a77230   18w  FIFO    0,8      0t0 78553492 pipe
lsof      21484 a77230   cwd   DIR   0,21     4096  9738253 /home/a77230/ESC/NPB3.3.1-MZ/NPB3.3-MZ-SER nas-0-0.local:/state/partition1/cpd-2019/a77230
lsof      21484 a77230   rtd   DIR    8,1     4096        2 /
lsof      21484 a77230   txt   REG    8,1   145872   802498 /usr/sbin/lsof
lsof      21484 a77230   mem   REG    8,1   156872   281686 /lib64/ld-2.12.so
lsof      21484 a77230   mem   REG    8,1  1922112   281687 /lib64/libc-2.12.so
lsof      21484 a77230   mem   REG    8,1    22536   281689 /lib64/libdl-2.12.so
lsof      21484 a77230   mem   REG    8,1   124624   281706 /lib64/libselinux.so.1
lsof      21484 a77230   mem   REG    8,1    65928   262179 /lib64/libnss_files-2.12.so
lsof      21484 a77230    4r  FIFO    0,8      0t0 78554022 pipe
lsof      21484 a77230    7w  FIFO    0,8      0t0 78554023 pipe
```

A.0.1 Graphs

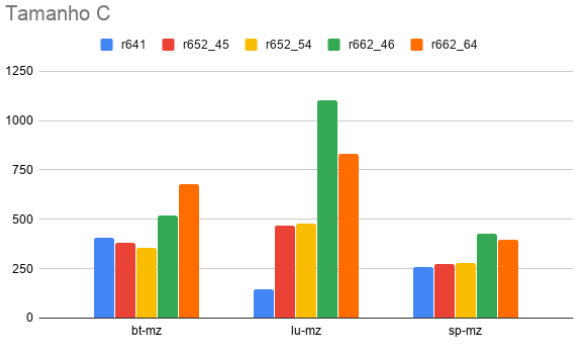
Tempo de execução



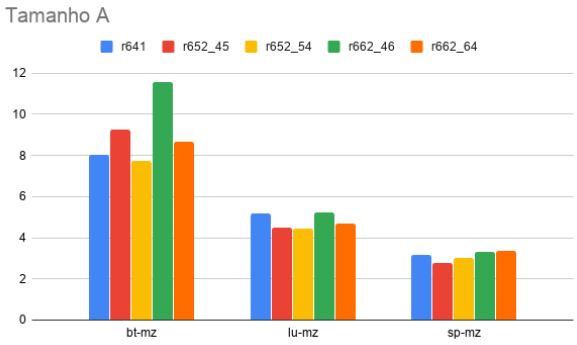
Tempo de Execução @ MPI, A



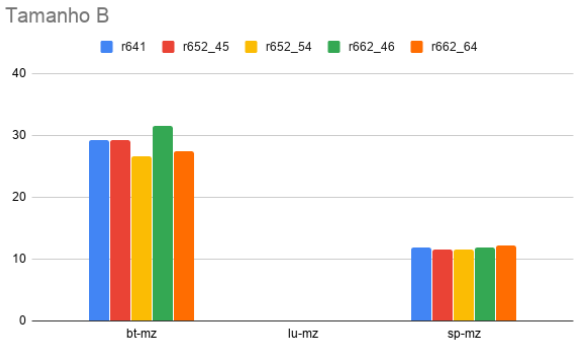
Tempo de Execução @ MPI, B



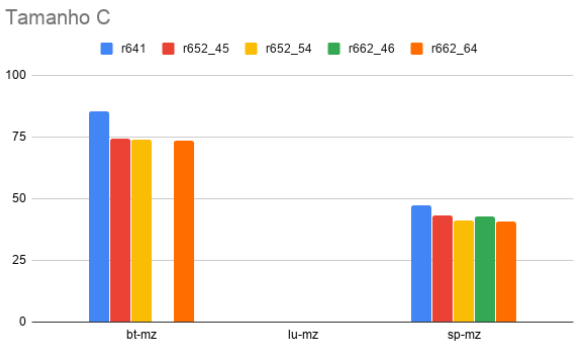
Tempo de Execução @ MPI, C



Tempo de Execução @ OMP, A

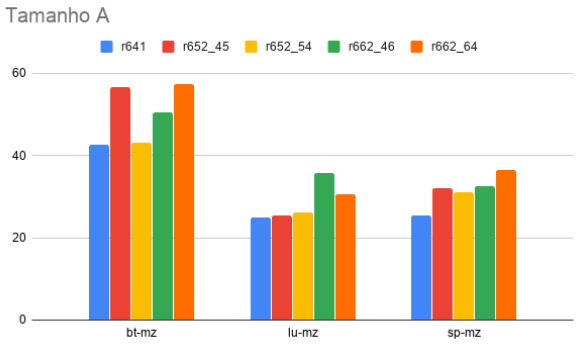


Tempo de Execução @ OMP, B

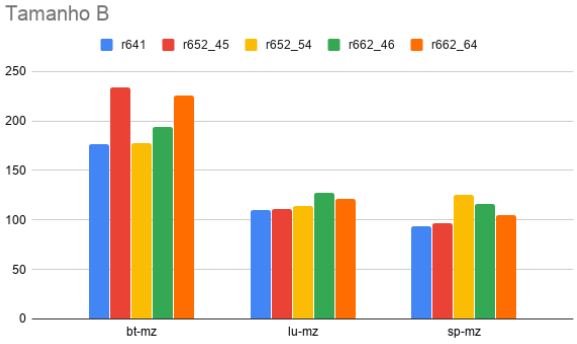


Tempo de Execução @ OMP, C

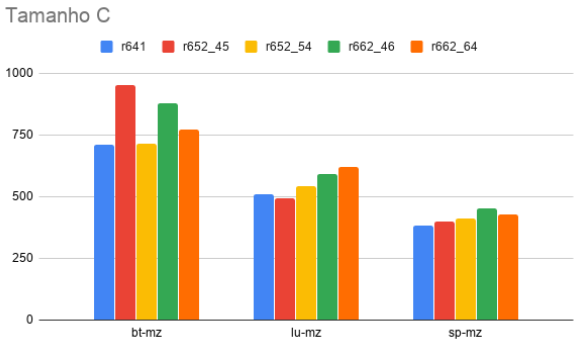
Used Memory



Tempo de Execução @ SER, A

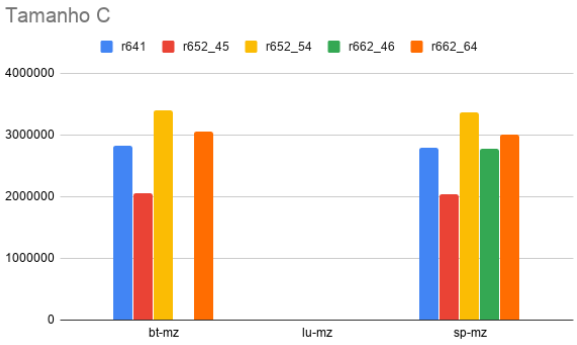


Tempo de Execução @ SER, B

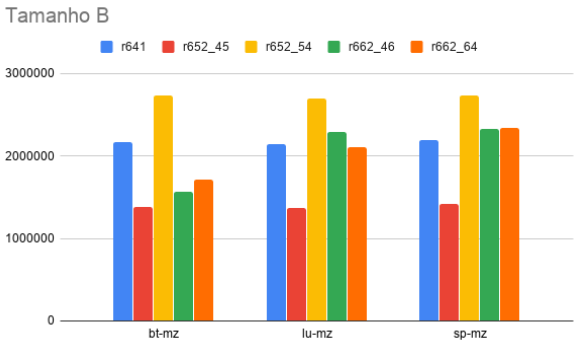


Tempo de Execução @ SER, C

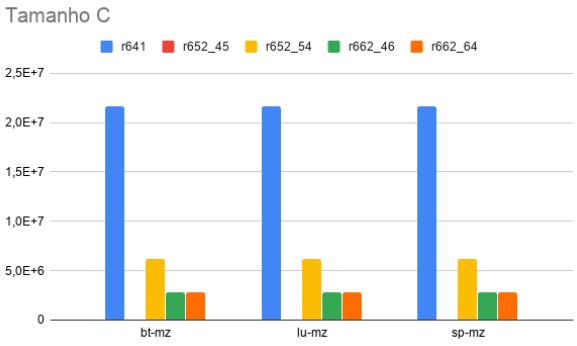
IO-WAIT



Used Memory @ OMP, C

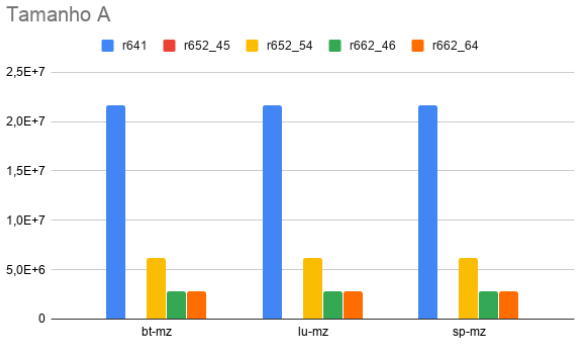


Used Memory @ SER, B

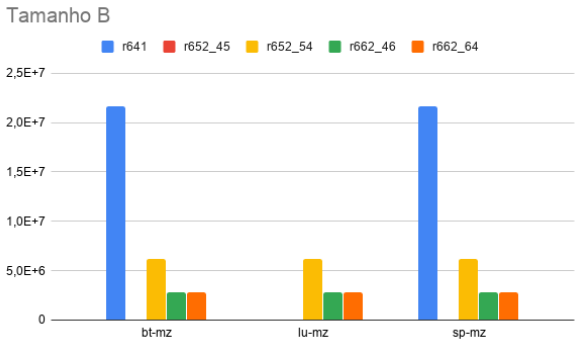


IO-wait @ MPI, C

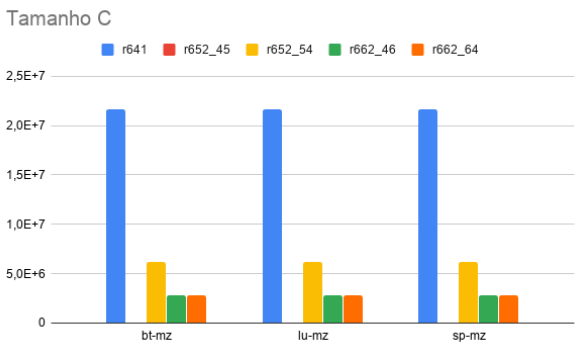
A.0.2 Resultados Obtidos (IDLE CPU)



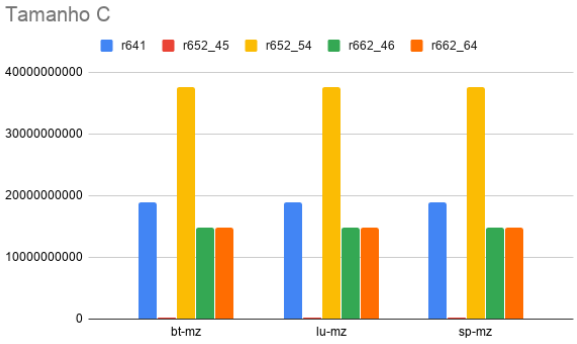
IO-wait @ SER, A



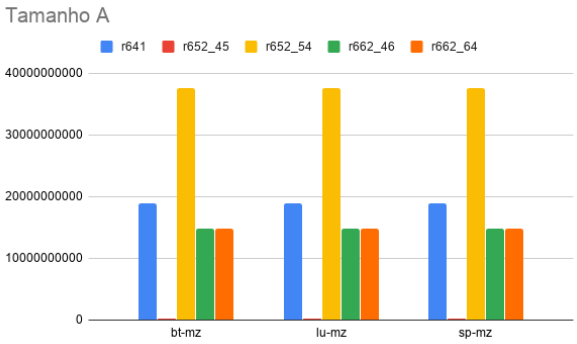
IO-wait @ MPI, B



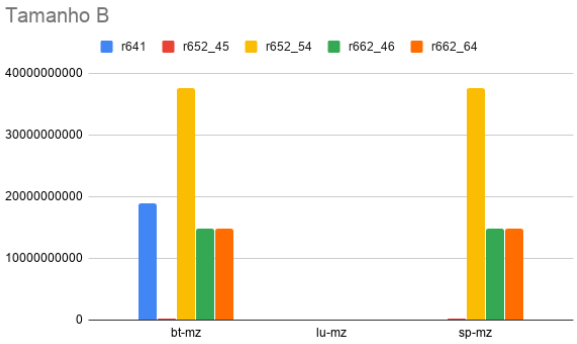
IO-wait @ SER, C



Idle CPU @ MPI, C



Idle CPU @ OMP, A



Idle CPU @ OMP, B