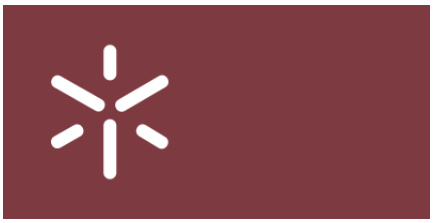

Universidade do Minho
Mestrado em Engenharia Informática

**Engenharia de Sistemas da Computação**
2018/2019 - Trabalho Prático 3

# *Benchmark* de *Input/Output* do sistema
## Ferramenta IOZONE

***Autores:***

| | |
|---|---|
| André Pereira | PG38923 |
| Vasco Leitão | PG38429 |

17 de Junho de 2019

# Conteúdo

# 1 Introdução

Este trabalho consistiu na realização da avaliação do sistema de ficheiros. Para isso foram utilizados as duas técnicas de *benchmarking*, isto é, *benchmark* ativo e passivo.

Para a realização do *benchmark* ativo foi utilizada a ferramenta **iozone**, esta utilitário, através do uso das opções indicadas, permitiu-nos realizar as medições do número de *bytes* escritos e lidos por segundo na escrita e leitura de ficheiros com diversos tamanhos, utilizando as chamadas de sistema *read* e *write* com tamanho de bloco diferentes.

Numa segunda fase deste trabalho foi feita a análise ativa da execução do mesmo aplicativo, de forma a atestar a veracidade, dos resultados anteriormente. Para esta fase foi utilizada a ferramenta **bpftrace**, de forma a armadilhar as chamadas ao sistema da aplicação anterior e assim obter informações que possibilitassem a obtenção de resultados semelhantes.

## 2 Monitorização

A monitorização foi feita com o recuso ao **iozone**, para obter os resultados este aplicativo foi corrido com as opções `i1` de forma a obter o número de *bytes* escritos e rescritos em ficheiro por segundo, foi também utilizada a opção `i1` para obter o número de *bytes* lidos e relido de ficheiro por segundo e por fim foi também utilizada a opção `i2` para obter os mesmos resultados mas com escritas e leituras de partes escolhidas aleatoriamente desse ficheiro.

Através da execução deste aplicativo foi obtido o seguinte *benchmark* passivo:

```
                                                    random    random
    kB  reclen    write  rewrite    read    reread    read     write
 16384    2048  2698817  3336900  3052190  2428695  2948346  1532657
```

Figura 1: Benchmark IOzone para uma quantidade de 16384 kB com chunks de 2048 kB

O *benchmark* anterior foi gerado através do comando `iozone -i0 -i1 -i2 -s16384 -r2048`, que possibilitou assim a realização dos seis testes. Foi calculada a velocidade em `kBytes/seg` para os três modos de escrita e os três modos de leitura. Para este teste, foi escolhido um bloco de 16384 kB, cujo foi lido e escrito totalmente de forma iterativa com *chunks* de 2048 kB.

Foi deduzido e pressuposto que a aplicação `IOzone` realiza os seguintes testes sequencialmente sem existir qualquer controlo de concorrência: `write - rewrite - read - reread - random_read - random_write`.

Por exemplo, para o teste realizado pressupõe-se que o primeiro conjunto de *writes* instanciados são relativos a um *write* "normal"de 16384 kB, que o próximo conjunto de *writes* seja para o teste *rewrite*, e assim sucessivamente.

### Script

Para atestar a veracidade dos resultados foi então criado um *script bpftrace* que obtivesse as mesmas informações. De seguida será explicado o processo realizado para a obtenção das informações relativas às escritas. Para recolher as informações necessárias relativas às escritas, *script* apenas necessita de fazer o *probing* de três chamadas de sistema, o *write*, o *open* e o *close*, sendo que para a *system call write* é necessário armadilhar tanto a entrada como o retorno da rotina.

A *script* começa então por inicializar, ao ser chamada a rotina *open*, ou no caso de já terem sido chamadas a esta rotina, as variáveis *block* e *time* com o valor zero, estas variáveis são responsáveis por armazenar, respetivamente, o número total de *bytes* e o tempo demorado na escrita de um ficheiro.

De seguida é apanhada a entrada rotina *write*, aqui será utilizada uma marca temporal para que se possa saber no final da execução o tempo que esta rotina demorou, guardando assim o momento que foi iniciada a escrita de um *chunk*, nesta fase é também lido o valor do argumento na posição dois da rotina invocada, este valor é o número de *bytes* que se pretende escrever no ficheiro, que

representa a escrita de apenas um *chunk* no mesmo. Assim no retorno desta rotina é somado à variável *block* o numero de bytes escritos por essa rotina, e também somado à variável *time* a diferença entre uma marca temporal obtida no instante que é invocado o retorno e a marca temporal obtida na entrada na rotina, ou seja, é somada o tempo que esta rotina demorou. Nesta fase é ainda utilizada uma variável *n_ chuks* que é incrementada, contabilizando assim quantas vezes esta rotina é chamada para cada ficheiro. De notar que na *probe* relativa à invocação da *system call write*, apenas são processadas as instruções anteriormente mencionadas, caso o número de bytes a ser escrito seja superior a 4000 kB, pois este é aproximadamente o número mínimo admitido para cada teste, e devido ao *bpftrace* intercetar múltiplas operações de escrita com valores muitos inferiores a este que não são usados para avaliar o sistema.

Por fim, ao ser invocada a *system call close* é realizado o cálculo da quantidade de *kbytes* escritos para o ficheiro por segundo, este cálculo obtido através da razão entre o número total de *bytes* escritos ao longo das múltiplas chamadas write (variável block) e a soma do tempo demorado por cada uma destass escritas, ou seja, a divisão da variável *block* pela variável *time*, de notar que a variável *block* é multiplicada por um escalar pois esta representa o número de *bytes* escritos e não o número de *kbytes*, e o tempo armazenado encontra-se em nanosegundos, por isso existe a necessidade desta conversão. De seguida é armazenado este valor num *array* associativo, isto é, guarda o valor desta variável em relação ao tamanho do bloco total escrito em ficheiro, ao tamanho de cada *chunck* escrito, à quantidade de *chunks* necessários e ao tipo do teste realizado. Mais uma vez, foi filtrado apenas os ficheiros superioes a 1000, pois foram intersetados mas não representam qualquer valor significativo para a avaliação do *I/O*. Para distinguirmos o tipo de escrita realizado é utilizada a variável *modo*, incrementando e ficando com o resto da divisão por três, isto deve-se ao facto de se medirem apenas três modos e seres testados ciclicamente, a escrita normal, a reescrita do ficheiro e a escrita de blocos em posições aleatórias. Para cada modo é utilizado um ficheiro diferente e portanto é feito o *open* e o *close* desse ficheiro. De notar por fim que foi utilizado um filtro em todos estes *tracepoints* em que o nome do executável fosse **iozone**, filtrando assim todas as *system calls* de modo a serem apenas inspecionadas aquelas instanciadas pelo `IOzone`.

O processo utilizado para a obtenção das informações relativas às leituras é análogo a este, isto é, apenas utilizando a *system call read* ao invés da *write*. A utilização de uma só *script* para a realização das medições relacionadas com escritas e com as leitura, levou à necessidade da utilização de *flags*, para se saber em qual dos casos se estava a processar. Estas *flags* são então ativas no retorno das funções *write* e *read*, para que no momento da execução das instruções no *tracepoint close*, sejam calculadas as métricas relativa ao conjunto de operações de escrita ou de leitura realizadas anteriormente.

Tal como na *probe* inserida no *write*, existe na *probe* do return do *read*, uma condição, que filtra apenas as leituras iguais ou superiores a 4 kB, isto deve-se a novamente o *bpftrae* intersetar *syscalls reads* em excesso provenientes da aplicação *IOzone*. Na interseção do *close* relativo a uma leitura, foram ainda descartadas as leituras de 0 kB (que existiram no sistema operativo Ubuntu), resultando a condição `if (@block>0)`. A condição `if (@mr>-1)` deve-se à exis-

tência de ser sempre realizada uma leitura de 4 kB no início da *benchmark*, ignorando assim esta chamada.

Essencialmente, a velocidade foi calculada, através do quociente entre o tamanho do ficheiro escrito e a soma de todas as chamadas do sistema *write* ou *read*. De notar que não realizado o quociente entre o tamanho do ficheiro e o tempo despendido entre um *open* e um *close*, pois este apresenta o tempo despendido pelo processador entra as escritas dos vários *chunks* e a latência do *CPU* e do disco nas operações realizadas.

## 2.1 Resultados

São então apresentados os resultados obtidos pela *script* criada, que avaliam a entrada e a saída de dados do sistema de ficheiros. Os valores provenientes da *script* apresentam sempre resultados muito próximos às métricas provenientes da *benchmark* do *IOZone*.

Na realização dos testes não foram utilizados ficheiros com tamanhos muitos baixos, como por exemplo 4 kB, pois para ficheiros muito pequenos as escritas e leituras são realizadas num tempo muito pequeno. Dado este tempo muito reduzido, os valores apresentados não serão coerentes pois quanto menor o tempo, maior será o ruído a que estará sujeito quer a *script* em *bpftrace*, quer o próprio *IOzone*. Ora, a resolução que o *IOzone* usa é de 0.000001 segundos, o *bpftrace* consegue medir o tempo em nanosegundos, acrescentando ainda o ruído que está sujeito, *benchmarks* com tamanhos inferiores não serão viáveis.

**Teste 1**

| kB | reclen | write | rewrite | read | reread | random read | random write |
|---|---|---|---|---|---|---|---|
| 16384 | 2048 | 2529631 | 1624735 | 3368964 | 2732079 | 3290562 | 1594870 |

Figura 2: IOZONE: Bloco de 16384 kB com chunks de 2048 kB

```
       kB, reclen, n_chunks, mode:   kb/s
@write[16384, 2048, 8, random_write]: 1646988
@write[16384, 2048, 8, rewrite]: 1672243
@write[16384, 2048, 8, write]: 2598125

@read[18432, 2048, 9, reread]: 2858698
@read[16384, 2048, 8, random_read]: 3437644
@read[18432, 2048, 9, read]: 3449036
```

Figura 3: Resultados obtidos pela script equivalentes à primeira benchmark

**Teste 2**

```
                                                          random    random
    kB  reclen    write  rewrite    read    reread    read     write
 32768    4096  2036651  1489727  3142625  3300900  5382416  3309564
```

Figura 4: IOZONE: Bloco de 32768 kB com chunks de 4096 kB

```
        kB, reclen, n_chunks, mode:   kb/s
@write[32768, 4096, 8, rewrite]: 1530335
@write[32768, 4096, 8, write]: 2095243
@write[32768, 4096, 8, random_write]: 3406036

@read[36864, 4096, 9, read]: 3278549
@read[36864, 4096, 9, reread]: 3438754
@read[32768, 4096, 8, random_read]: 5617526
```

Figura 5: Resultados obtidos pela script equivalentes à segunda benchmark

A *script* imprime para o *output* dois *arrays* associativos. Um referente aos 3 tipos de escritas e outro referente às 3 diferentes leituras. Assim as chaves do array são: o tamanho do ficheiro, o tamanho do *chunk*, o número de *chunks*, o modo da leitura/escrita; o valor do *array* representa a velocidade em *kbps* do teste realizado.

De seguida, é apresentada uma tabela que demonstra facilmente a grande semelhança entre os resultados obtidos nos dois testes realizados, pela *script* e pela aplicação de monitorização.

| **Teste** | **App** | **kB** | **reclen** | **write** | **rewrite** | **read** | **reread** | **rand read** | **rand write** |
|---|---|---|---|---|---|---|---|---|---|
| 1 | IOZone | 16384 | 2048 | 2529631 | 1624735 | 3368964 | 2732079 | 3290562 | 1594870 |
| 1 | Script | 16384 | 2048 | 2598125 | 1672243 | 3449036 | 2858698 | 3437644 | 1646988 |
| 2 | IOZone | 32768 | 4096 | 2036651 | 1489727 | 3142625 | 3300900 | 5382416 | 3309564 |
| 2 | Script | 32768 | 4096 | 2095243 | 1530335 | 3278549 | 3438754 | 5617526 | 3406036 |

Tabela 1: Comparação de resultados apresentados pelo IOzone e pela script

# 3 Conclusão

Através do desenvolvimento desta *script*, foi possível realizar um pouco de engenharia reversa sob a aplicação de monitorização *IOzone* utilizada, e assim observar o comportamento que a aplicação exerce sob o *kernel do sistema* através da observação das *syscalls* invocadas por este programa, decifrando assim que conjunto de operações que são realizadas para efetuar o *benchmark* do I/O.

Houve, no entanto pequenos aspetos e comportamentos inesperados, que foram observadas ao longo do desenvolvimento desta *script* em duas máquinas com sistemas operativos diferentes: MacOS e Ubuntu. No computador Mac, foi utilizada a linguage de *script DTrace* que apresentou números de *writes* exercidas no sistema supostas aquelas que se estava à espera: $((tam_{bloco}/num_{chunks}) \times 3)$, por *benchmark*. Por outro lado, no linux utilizado, o *BPFTrace* apresentou números um pouco diferentes, existindo alguns *writes* que não eram espectáveis. Quanto à quantidade de *reads* também existem valores um pouco controversos, quer no Mac, quer no Linux, pois na leitura de um determinado bloco de tamanho $x$ kB, utilizando $n$ chunks, o número de chunks lidos por ficheiro foi sempre $n + 1$, ou seja, na esscrita de um ficheiro de tamanho $x$, depois da *syscall open()* e antes da *close()*, existe sempre mais um *chunk* de $x/n$ kB que não era espectável.

Provavelmente, o número de *writes* em excesso que foram captados pelo *bpftrace* são relativos à informação que é enviada para o *stdout* pela aplicação *IOzone*.

# 4 Anexo

## 4.1 Script de monitorização do I/O

```
BEGIN
{
    @ts = 0; @kbps = 0;
    @time = 0;
    @bytes = 0;
    @block = 0;
    @n_chunk = 0;

    @mw=0;
    @modew[0]="write";
    @modew[1]="rewrite";
    @modew[2]="random_write";
    @mr=-1;
    @moder[-1]="error";
    @moder[0]="read";
    @moder[1]="reread";
    @moder[2]="random_read";
}


tracepoint:syscalls:sys_exit_openat
/comm == "iozone"/
{
    @time=0;
    @block=0;
}

tracepoint:syscalls:sys_enter_read
/comm == "iozone"/
{
    @ts = nsecs;
    @bytes = args->count;
}

tracepoint:syscalls:sys_exit_read
/comm == "iozone"/
{
  if (@bytes > 4000) {
      @block=@block+@bytes;
      @n_chunk++;
      @time=@time+(nsecs - @ts);
      @flag_read=1;
      @flag_write=0;
  };
}

tracepoint:syscalls:sys_enter_write
/comm == "iozone"/
{
```

```
    @ts = nsecs;
    @bytes = args->count;
}

tracepoint:syscalls:sys_exit_write
/comm == "iozone"/
{
  if (@bytes > 4000) {
      @block=@block+@bytes;
      @n_chunk++;
      @time=@time+(nsecs - @ts);
      @flag_read=0;
      @flag_write=1;

  };
}

tracepoint:syscalls:sys_enter_close
/comm == "iozone"/
{
  if (@block>0 && @flag_read==1) { //se antes do close houve
      uma leitura
      @kbps = (@block * 1000000) / @time;
      if (@mr>-1) {
        @read[@block/1024, @bytes/1024, @n_chunk, @moder[@mr]]
            = @kbps;
      };
      @mr++;
      @mr=@mr%3;
  };

  if (@block>1000 && @flag_write==1) { se antes do close houve
      uma escrita
      @kbps = (@block * 1000000) / @time;
      @write[@block/1024, @bytes/1024, @n_chunk, @modew[@mw]]
          = @kbps;
      @mw++;
      @mw=@mw%3;
  };

  @time=0;
  @block=0;
  @n_chunk=0;
  @kbps=0;
}


END
{
    printf("\n        kB, reclen, n_chunks, mode:    kb/s\n");
    print(@write);
    print(@read);
}
```