

**Mestrado em Engenharia Informática**

*Engenharia dos Sistemas de Computação - 2014/2015*

Análise de Traçados com *strace*

14 de Maio de 2015

**Resumo**

Este trabalho resultará num relatório de desempenho equivalente ao que é produzido diretamente pela aplicação iozone e produzir gráficos que representem os padrões temporais de acesso aos dados, para as operações de leitura/escrita e busca (lseek).

**Fábio Gomes** pg27752

# **Índice**

[**Índice** 2](#_Toc419403335)

[Introdução 3](#_Toc419403336)

[Strace e Parâmetros de teste 4](#_Toc419403337)

[Estatísticas 5](#_Toc419403338)

[*Tempo* 5](#_Toc419403339)

[*Total de Operações E/S* 5](#_Toc419403340)

[*Banda utilizada pelo write* 6](#_Toc419403341)

[*Tamanho de Blocos de Ficheiros no write* 6](#_Toc419403342)

[*Sumário do write* 7](#_Toc419403343)

[*Banda utilizada pelo read* 8](#_Toc419403344)

[*Tamanho de Blocos de Ficheiros no read* 8](#_Toc419403345)

[*Sumário do read* 9](#_Toc419403346)

[*Estatísticas dos Ficheiros Criados* 9](#_Toc419403347)

[Conclusão 9](#_Toc419403348)

# Introdução

O *strace* é uma ferramenta para a depuração de programas cujos traçados quando filtrados e analisados podem também ser usados para estudar o padrão de execução das aplicações. Monitoriza interações entre os programas e o kernel do Linux, como chamadas ao sistema, sinais e mudanças no estado do processo. O trabalho do *strace* só é possível devido ao *ptrace*.

Com a ajuda do ficheiro *refazStFd.py* disponibilizado pelo professor foi possível obter melhores informações sobre a utilização dos vários ficheiros temporários criados pelo *iozone*.

# Strace e Parâmetros de teste

Para este trabalho utilizei o strace versão 4.5.19 instalada no cluster. Primeiramente foi necessário utilizar o *strace* da seguinte forma num job:

strace -T -ttt -o strace.out /opt/iozone/bin/iozone -R -i0 -i1 -i2 -i5 -g 1G

Com o ficheiro output *strace.out* passei-o pelo *refazStFd.py* para tratar dos ficheiros temporários.

/share/apps/IOAPPS/refazStFd.py < strace.out > ref.rsf

O *ref.rsf* já está tratado e então é altura de o passar pelo *strace\_analyzer* para gerar algumas estatísticas para melhor compreender o que se passou naquela execução do *iozone*.

/share/apps/IOAPPS/strace\_analyzer\_ng\_0.09.pl ref.rsf > stanREF.txt

# Estatísticas

Tendo já o ficheiro pronto para análise, é possível obter as seguintes informações.

## *Tempo*

O programa na totalidade demorou cerca de 1.32 segundos a concluir, sendo que 65.36% desse tempo foi em operações de E/S ao sistema perfazendo 0.86 segundos.

|  |  |
| --- | --- |
|  |  |
| **Elapsed Time for run** | **1.322435 (secs)** |
| Total IO Time | 0.864388 (secs) |
| Total IO Time Counter | 1653 |
| Percentage of Total Time | 65.363369% |

## *Total de Operações E/S*

As seguintes chamadas ao sistema via operações de E/S foram contabilizadas da seguinte forma:

|  |  |
| --- | --- |
| Command | Count |
| access | 1 |
| lseek | 387 |
| stat | 56 |
| unlink | 2 |
| open | 13 |
| close | 15 |
| creat | 1 |
| fstat | 6 |
| fsync | 12 |
| read | 519 |
| write | 567 |

Como se trata de um teste de escrita em disco, é normal que os comandos *lseek*, *read* e *write* sejam os mais utilizados.

## *Banda utilizada pelo write*

Com o histórico da chamada ao sistema write, foi possível gerar o seguinte gráfico onde podemos analisar a quantidade de informação escrita por segundo (*Banda*/*Bandwidth*) ao longo da execução. Os blocos são de 4096 Bytes.

## *Tamanho de Blocos de Ficheiros no write*

Os ficheiros escritos foram maioritariamente utilizados com tamanhos compreendidos entre 1 KB e 8 KB, nomeadamente os 4096 Bytes (4KB) mencionados no teste acima. O número elevado do primeiro intervalo deve-se a pequenas escritas que ocorrem ao longo da execução.

|  |  |
| --- | --- |
| IO Size Range | Number of syscalls |
| 0KB < < 1KB | 183 |
| 1KB < < 8KB | 384 |
| 8KB < < 32KB | 0 |
| 32KB < < 128KB | 0 |
| 128KB < < 256KB | 0 |
| 256KB < < 512KB | 0 |
| 512KB < < 1000KB | 0 |
| 1000KB < < 10MB | 0 |
| 10MB < < 100MB | 0 |
| 100MB < < 1GB | 0 |
| 1GB < < 10GB | 0 |
| 10GB < < 100GB | 0 |
| 100GB < < 1TB | 0 |
| 1TB < < 10TB | 0 |

## *Sumário do write*

Concluídas as análises anteriores à chamada *write* é possível tirar uma conclusão geral e algumas estatísticas interessantes com os valores obtidos.

|  |  |
| --- | --- |
| -- WRITE SUMMARY -- |  |
| Total number of Bytes written | 1,575,255 (1.575255 MB) |
| Number of Write syscalls | 567 |
|  | |
| Average (mean) Bytes per syscall | 2,778.22751322751 (Bytes) |
| Standard Deviation | 1,908.92071764713 (Bytes) |
| Mean Absolute Deviation | 2,093.08630644254 (Bytes) |
| Median Bytes per syscall | 4,096 (Bytes) |
| Median Absolute Deviation | 1,317.77248677249 (Bytes) |
|  | |
| Time for slowest write syscall (secs) | 0.003611 |
| Line location in file | 190 |
|  | |
| Smallest write syscall size | 1 (Byte) |
| Largest write syscall size | 4096 (Bytes) |

No total foi escrito 1.575 MB, num total de 567 chamadas ao *write*, como mostra a primeira estatística (*Total de Operações E/S*). Como a maior parte do teste se esteve pelos 4096 Bytes de bloco, então foi essa a Mediana nos Bytes lidos por chamada.

## *Banda utilizada pelo read*

Com o histórico da chamada ao sistema read, foi possível gerar o seguinte gráfico onde podemos analisar a quantidade de informação lida por segundo (*Banda*/*Bandwidth*) ao longo da execução. Os blocos são de 4096 Bytes.

## *Tamanho de Blocos de Ficheiros no read*

Os ficheiros escritos foram maioritariamente utilizados com tamanhos compreendidos entre 1 KB e 8 KB, nomeadamente os 4096 Bytes (4KB) mencionados no teste acima.

|  |  |
| --- | --- |
| IO Size Range | Number of syscalls |
| 0KB < < 1KB | 3 |
| 1KB < < 8KB | 516 |
| 8KB < < 32KB | 0 |
| 32KB < < 128KB | 0 |
| 128KB < < 256KB | 0 |
| 256KB < < 512KB | 0 |
| 512KB < < 1000KB | 0 |
| 1000KB < < 10MB | 0 |
| 10MB < < 100MB | 0 |
| 100MB < < 1GB | 0 |
| 1GB < < 10GB | 0 |
| 10GB < < 100GB | 0 |
| 100GB < < 1TB | 0 |
| 1TB < < 10TB | 0 |

## *Sumário do read*

Concluídas as análises anteriores ao *read* é possível tirar uma conclusão geral e algumas estatísticas interessantes com os valores obtidos.

|  |  |
| --- | --- |
| -- READ SUMMARY -- |  |
| Total number of Bytes read | 2,113,454 (2.113454 MB) |
| Number of Read syscalls | 519 |
|  | |
| Average (mean) Bytes per syscall | 4,072.16570327553 (Bytes) |
| Standard Deviation | 262.594685187763 (Bytes) |
| Mean Absolute Deviation | 3,809.57101808775 (Bytes) |
| Median Bytes per syscall | 4,096 (Bytes) |
| Median Absolute Deviation | 23.8342967244701 (Bytes) |
|  | |
| Time for slowest read syscall (secs) | 0.000236 |
| Line location in file | 1435 |
|  | |
| Smallest read syscall size | 832 (Bytes) |
| Largest read syscall size | 4096 (Bytes) |

No total foi lido perto de 2.1 MB, num total de 519 chamadas ao *read*, como mostra a primeira estatística (*Total de Operações E/S*). Como a maior parte do teste esteve pelos 4096 Bytes de bloco, então foi essa a Mediana nos Bytes escritos por chamada.

## *Estatísticas dos Ficheiros Criados*

Ao todo foram criados 7 ficheiros temporários e a quantidade de informação utilizada foi a seguinte:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Read Bytes | Avg. Bytes/sec | Write Bytes | Avg. Bytes/sec |
| iozone.tmp\_0 | 0 | 41,386,883.12 | 524,288 | 76,985,853.54 |
| iozone.tmp\_1 | 0 | 0 | 524,288 | 59,141,200.83 |
| iozone.tmp\_2 | 528,384 | 73,521,918.19 | 0 | 0 |
| iozone.tmp\_3 | 528,384 | 99,184,372.43 | 0 | 0 |
| iozone.tmp\_4 | 524,288 | 120,853,261.23 | 0 | 0 |
| iozone.tmp\_5 | 0 | 0 | 524,288 | 82,099,558.43 |
| iozone.tmp\_6 | 524,288 | 108,666,288.41 | 0 | 0 |

# Conclusão

Com a ferramenta *strace* foi possível ter uma perspetiva diferente de como o *iozone* trabalha. Analisando de um ponto de vista de chamadas ao sistema via *syscalls* como *lseek*, *write* ou *read*.