Este trabalho tem em vista a familiarização com os instrumentos e técnicas dos sistemas de operação modernos, para a análise de desempenho de uma aplicação. Os enunciados que seguem, incidem sobre o uso do PERF, uma ferramenta de análise de desempenho no Linux, baseado em perf_events, disponível no kernel do Linux.

1. Análise de uma aplicação C/C++

Considere o programa *sort*, cujo código fonte em *prog_sort.tgz* está disponível na pasta **Enunciados dos TP** da secção de conteúdos da BlackBoard.

a) Use a ferramenta **perf** para obter e comparar #I (número de instruções), #MISS, etc.., e encontrar razões que expliquem as diferenças de desempenho das variantes do algoritmo de ordenação.

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

```
// sort 2
perf stat -e instructions,cache-misses ./sort 2 1 100000000
```

(repetir para as outras variantes)

b) Criar e analisar o perfil de execução para identificar o tipo requisitos de cada variante do algoritmo (percentagem de tempo em: chamadas a bibliotecas, módulos do kernel, funções do utilizador, etc).

```
// sort 1
perf record -F 99 ./sort 1 1 100000000
perf report -n --stdio
```

(repetir para as outras variantes)

c) Usar os "*Flame Graph*" para identificar cada um dos algoritmos com base no aninhamento das chamadas (e.g., identificando recursividade, etc..)

```
// sort 1
perf record -F 99 -ag ./sort 1 1 100000000
perf script | ./stackcollapse-perf.pl | ./flamegraph.pl> sort.svg
```

(repetir para as outras variantes)

d) [*opcional*] Identificar eventuais melhorias em sort2 analisando o perfil de execução ao nível do assembly

```
perf record -g ./sort 2 1 100000000
perf annotate --stdio
```

(repetir para as outras variantes)

2. Á procura dos pontos quentes de uma aplicação

O método a utilizar tem início com a **recolha dados** que acompanham a execução de uma aplicação, desde o código-fonte, às chamadas ao sistema, ao próprio *kernel*, incluindo os valores dos contadores de eventos de *hardware/software* disponíveis.
Com base naqueles dados, podem fazer-se medições básicas do comportamento da aplicação para determinar e quantificar problemas de desempenho e, posteriormente, responder a questões tais como: quais os pontos quentes de uma aplicação, quais as transações que estão a ocorrer e com que tempo de atraso, quanto tempo e porque razão uma aplicação está *on/off* no CPU ?.

O desenvolvimento deste trabalho compreende as seguintes fases:

1. Estudar detalhadamente o tutorial (ver abaixo) que utiliza a ferramenta **PERF** para a procura dos pontos quentes de uma aplicação de multiplicação de matrizes
2. Repetir aquele tutorial, adaptando-o ao ambiente de desenvolvimento do SeARCH, documentando e discutindo os resultados obtidos pela execução das aplicações propostas.
3. Geração de gráficos considerados necessários com base na ferramenta **Flame Graphs**.

Nota1: Qualquer nó de computação do cluster poderá ser usado. No entanto, os que possuem caraterísticas físicas (contadores) mais parecidos com os usados no tutorial, são compute-431-x e compute-432-x, acessíveis especificando r431/342 (e.g. `qsub -q day -V -I -lnodes=1:r431`).
Nota2: O tamanho do problema no tutorial `#define MSIZE 500` deverá ser ajustado convenientemente, para refletir as capacidades físicas do nó de computação usado.

Material disponível

PERF - Tutorial em 3 Partes

- 1 - Parte: Procura dos pontos quentes de uma aplicação em execução
- 2 - Parte: Contagem de eventos de Hardware
- 3 - Parte: Perfis de Eventos de Hardware
- `naive.c`: Código na pasta **Enunciados dos TP** da secção de conteúdos da BB.

Ferramenta de geração de gráficos

- FlameGraphs