UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

EDUARDO LUIZ SCHADE SOARES

CÁLCULO DE APROXIMAÇÃO DA RAIZ DE 2 E DE TEMPO DE CHAVEAMENTO DE PROCESSOS

PONTA GROSSA
2018

- **Introdução**

A série utilizada para a demonstração do tempo de chaveamento entre processos foi a raiz de 2. Através desse cálculo foi possível "consumir" processamento da CPU e obter resultados relacionados ao tempo utilizado para *tarefas administrativas*, resultado da mudança de contexto dos processos.

- **Série Iterativa**

A raiz de 2 é um número irracional, ou seja, não é possível encontrar dois números inteiros que divididos um pelo outro resultem em $\sqrt{2}$. Assim, utiliza-se a aproximação para chegar ao resultado, cerca de *1,41421*.

O cálculo de aproximação utilizado para chegar ao resultado foi:

$$X_{n+1} = \frac{Xn}{2} + \frac{1}{Xn}$$

Adaptado para o código em C:

```
i = 1;
while (i <= iteracoes) {
      xn = x;
      x = xn/2 + 1/xn;
      i++;
}
```

Esse cálculo recursivo gera uma sequência tal como essa:

$$1;\ \frac{3}{2};\ \frac{17}{12};\ \frac{577}{408};\ \frac{665857}{470832}\ \ldots$$

O resultado equivalente a essas frações em decimais:

1; 1.5; 1.416666667; 1.414215686; 1.414213562;

A precisão dos resultados são proporcionais a quantidade de iterações, quanto mais iterações, mais próximo o resultado será do seu valor real.

- **Resultados do tempo de execução**

Foi criado um programa que gerava processos filhos com o cálculo da raiz de 2. Através da criação de mais processos, pôde-se fazer uma estimativa do tempo de chaveamento entre processos.

Para chegar aos resultados estimados do tempo de mudança entre os processos, foi realizada a execução do programa em diferentes cenários. Cada cenário equivale a execução do programa sendo ele equivalente a somente um processo, ou sendo ele pai de processos filhos. Cada cenário foi executado 5

vezes, e assim, chegou-se a um resultado médio para ser utilizado. Todos os tempos são em segundos.

Para a criação de processos dentro de um programa, foi utilizada uma função da linguagem C chamada *fork*.

O número de iterações para o cálculo da raiz de 2 foi fixado em *999999999* em todos os cálculos de tempo.

Segue abaixo a tabela de cálculos dos tempos médios em cada cenário.

CENÁRIO 1 - Execução sem forks (Referência)

Referência 1	14.469911
Referência 2	14.652099
Referência 3	14.711079
Referência 4	14.50477
Referência 5	14.526115
TEMPO MÉDIO	14.5727948

O cenário 1 equivale ao tempo que será utilizado como referência para o tempo de execução estimado com 1, 2, 3 e 4 forks. Logo, foi multiplicado o tempo de referência pela quantidade de processos que serão contabilizados. Sendo assim:

1 PROCESSO (0 Forks)	14.5727948
2 PROCESSOS (1 Forks)	29.1455896
3 PROCESSOS (2 Forks)	43.7183844
4 PROCESSOS (3 Forks)	58.2911792
5 PROCESSOS (4 Forks)	72.863974

CENÁRIO 2 - Execução com 1 fork (2 processos)

Referência 1	15.340753
Referência 2	15.464887
Referência 3	15.370589
Referência 4	16.081868
Referência 5	15.970999
TEMPO MÉDIO	15.6458192

CENÁRIO 3 - Execução com 2 forks (3 processos)

Referência 1	17.808901
Referência 2	18.310503
Referência 3	18.125062
Referência 4	18.295931
Referência 5	18.271655
TEMPO MÉDIO	18.1624104

CENÁRIO 4 - Execução com 3 forks (4 processos)

Referência 1	19.606588
Referência 2	20.171325
Referência 3	19.106218
Referência 4	18.954131
Referência 5	19.190398
TEMPO MÉDIO	19.405732

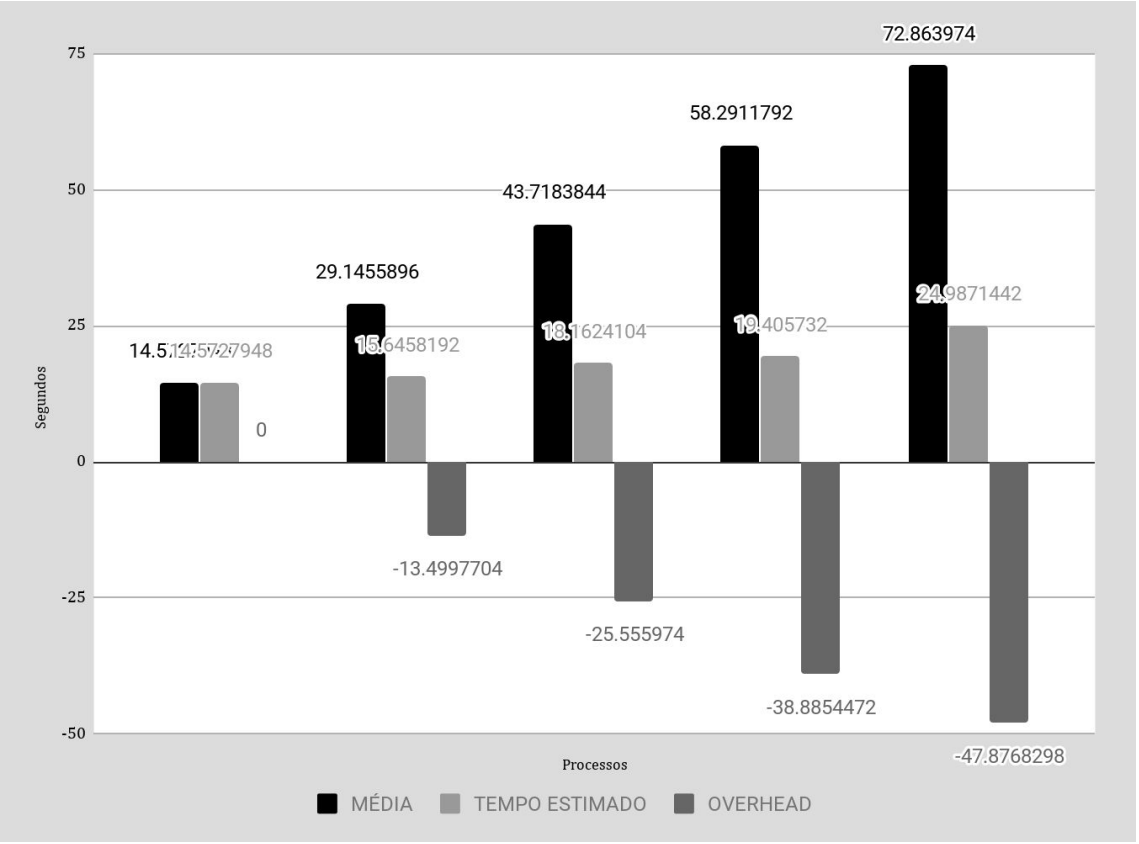
CENÁRIO 5 - Execução com 4 forks (5 processos)

Referência 1	23.924552
Referência 2	25.54719
Referência 3	25.009769
Referência 4	25.714285
Referência 5	24.739925
TEMPO MÉDIO	24.9871442

Com o cálculo estimado de tempo de execução em cada cenário, é finalmente possível encontrar o tempo de *overhead* através da subtração do tempo médio em cada cenário (2, 3, 4 e 5) pelo tempo de referência, encontrado no cenário 1. Assim:

	TEMPO DE REFERÊNCIA	TEMPO MÉDIO	OVERHEAD
1 PROCESSO (0 Forks)	14.5727948	14.5727948	0
2 PROCESSO (1 Forks)	29.1455896	15.6458192	-13.4997704
3 PROCESSO (2 Forks)	43.7183844	18.1624104	-25.555974
4 PROCESSO (3 Forks)	58.2911792	19.405732	-38.8854472
5 PROCESSO (4 Forks)	72.863974	24.9871442	-47.8768298

Traçando o gráfico, teremos:



O overhead resultou em um valor negativo, devido ao tempo de execução ser menor do que o tempo estimado.