INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

EP2 - MAC0438

João Marco Maciel da Silva 7577598

25 de Maio de 2015

1 CONFIGURAÇÃO DA MÁQUINA

Informações extraídas por *dmidecode, uname* e *lsb_release*.

Baseboard	Type	Laptop
	Manufacturer	SAMSUNG ELECTRONICS CO., LTD.
	Product Name	R480/R431/R481
Processor	Family	Core i3
	Max Speed	3200MHz
	Core Count	2
	Thread Count	4
	Characteristics	64-bit capable
Memory	Type	DDR3
	Data Width	64 bits
	Speed	667MHz
	Number of Devices	2
	Size	2048MB x2
Cache Size	L1	32 kB
	L2	256 kB
	L3	3072 kB
uname	OS	Linux
	Kernel Release	3.19.5-100.fc20.x86_64
	Processor type	x86_64
lsb_release	Description	Fedora 20

Os testes foram todos feitos em um terminal sem interface gráfica

2 TESTES

2.1 MEDIÇÃO DOS TEMPOS

Os tempos foram medidos com o *script* que acompanha o EP.
A variável *TIMEFORMAT* foi configurada para que o comando *time* retornasse os dados de tempo em um formato fácil de manipular.
Do comando *time* foi retornado o tempo de usuário, tempo de sistema, tempo real e percentual de CPU usado.

2.2 PARÂMETROS E REPETIÇÕES

Para cada entrada foram executadas 30 repetições.
Esse tamanho de amostra é um bom tamanho mínimo para que as médias das amostras possam ser aproximadas por uma normal satisfatoriamente.

2.2.1 1º PARÂMETRO - NÚMERO DE PROCESSOS

O EP foi testado nessa máquina com até 150 processos, porém o tempo execução para mais que 15 processos é muito alto e não possível retirar medidas estatísticas em tempo viável.
Para as medidas que estão nos gráficos foram feitos testes com até 10 processos.

2.2.2 2º PARÂMETRO - f OU m

Para cada conjunto de testes foram feitos testes com f e com m.

2.2.3 3º PARÂMETRO - PRECISÃO

Foram feitos testes com precisão de 10^{-10}, 10^{-100}, 10^{-1000}, 10^{-10000} e 10^{-10000}, tanto para f quanto para m.

2.2.4 4º PARÂMETRO - VALOR DE x

Foram usados quatro valores de x, $x = 1$, $x = 3,14159265359$, $x = 0.1$ e $x = 0$.

2.2.5 5º PARÂMETRO - S, D OU NADA

Todos os testes sem o 5º argumento também foram executados com o argumento d e quando não variando no 1º argumento também com o argumento s.

2.3 Geração dos gráficos

Os gráficos e cálculos estatísticos foram feitos com ajuda do Google Spreadsheets e os dados originais podem ser encontrados aqui.

2.4 Resultados

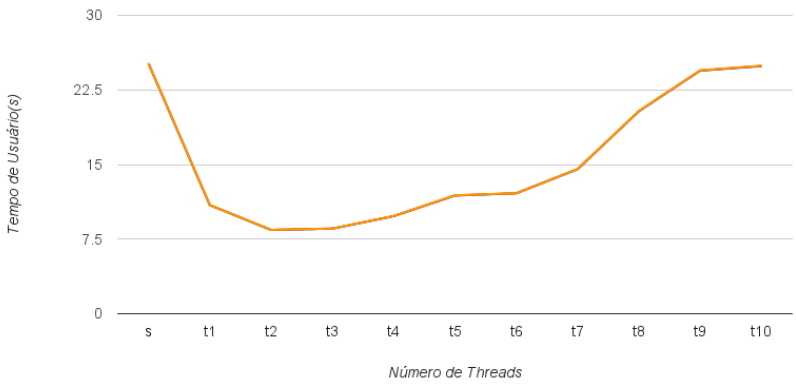

Figura 2.1: Média do tempo para x = 3.141592654, f = 1000

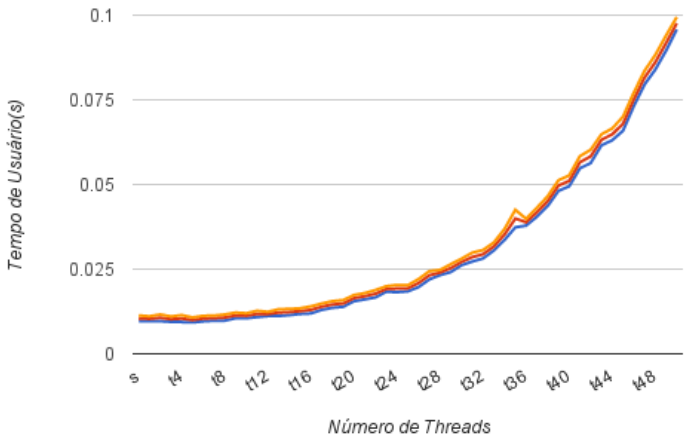

Figura 2.2: Média do tempo para x = 0, f = 100000

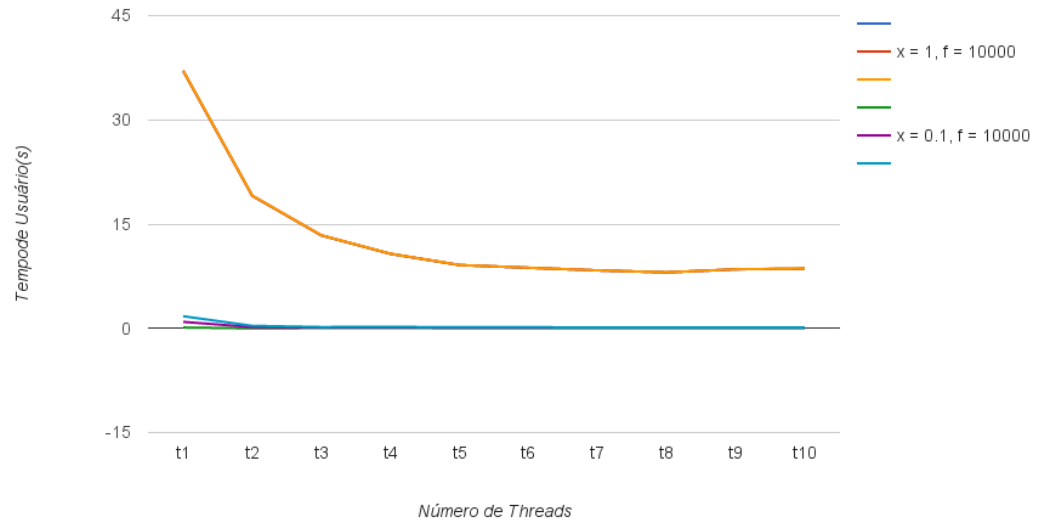

Figura 2.3: Médias do tempo para x = 1 e x = 0.1, f = 10000

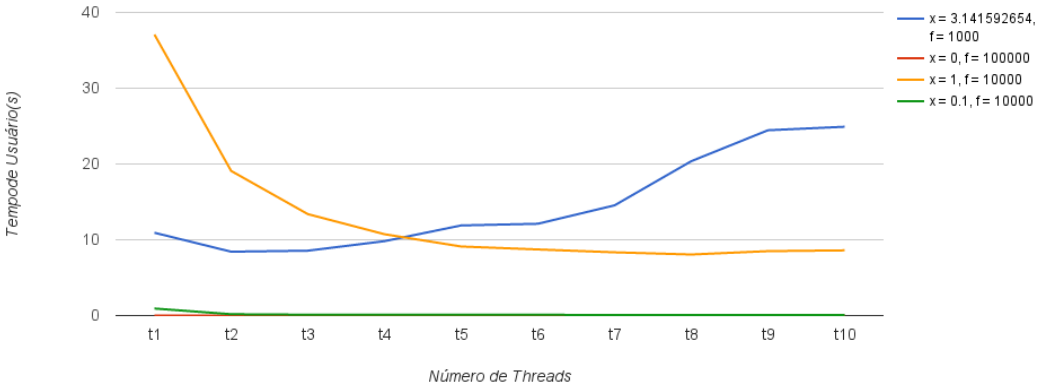

Figura 2.4: Compração entre as médias

O gráfico 2.1 mostra o formato esperado do gráfico do número de processos pelo tempo. É esperado esse formato pois o tempo diminui até uma certa quantidade de processos e depois há um *overhead* para gerenciar os processos e há um excesso de cálculos, que por si só são

custosos. É esperado que o tempo de *s* seja muito maior que *t1* e *t2*, pois não há nenhuma concorrência nesse caso.

O gráfico 2.2 mostra o formato esperado do gráfico do número de processos pelo tempo. Como $x = 0$ os termos são todos iguais a 0, logo ela acaba sempre depois da primeira rodada. Então espera-se que o gráfico seja sempre crescente. As linhas coloridas ao redor mostram o intervalo de confiança para uma confiança de 99%, no gráfico 2.1 essas linhas se fundem e não é possível verificá-las.

O gráfico 2.3 mostra as curvas de dois valores, um decimal e outro inteiro. A diferença entre os dois é, possivelmente, por conta do decimal ser menor e precisar de menos turnos para chegar ao valor da precisão. No caso de 10 processos e $f = 10000$, $x = 1$ precisou de 18 rodadas e $x = 0.1$ precisou de 14. É possível verificar que o formato da curva é o mesmo, mas que a área dos dois é completamente diferente.

A última conclusão que pode ser retirada com os resultados é que a curva para um x decimal com mais casas decimais deve inverter a derivada da cruva antes de um x com menos casas decimais. Esta conclusão pode ser percebida no gráfico 2.4, onde a curva com $x = 3.141592654$ inverte já com q igual a 4, enquanto os outros com $q > 10$.

3 Barreiras utilizadas

Foram utilizadas duas barreiras diferentes. A primeira, criada por mim, usa-se de uma exclusão mútua e um contador. Removendo o que é específico da aplicação seria:

```
type Barreira func()
func newBarreira(q int) Barreira {
  mutex := &sync.Mutex{}
  cond  := sync.NewCond(mutex)
  count := 0

  return func() {
    mutex.Lock()
    count++
    if count == q+1 {
      count = 0
      cond.Broadcast()
    } else {
      cond.Wait()
    }
    mutex.Unlock()
  }
}
```

Primeiro ele inicializa a barreira com tamanho q+1 (o +1 é para incluir o caminho padrão) e cada processo que chega na barreira chama a função da barreira. Dentro da função ele entra em uma área de exclusão mútua, incrementa o contador e se foro último lança uma

mensagem broadcast para todos que estiverem esperando, caso contrário apenas espera. No final libera a trava.
Esse código na sintaxe utilizada na aula é aproximadamente equivalente a:

```
sem mutex = 1
int count = 0
boolean cond = false

barreira {
  P(mutex)
  count++
  if (count == q+1) {
    count = 0
    cond = true
  } else {
    < V(mutex); await(cond) >
    P(mutex)
  }
  V(mutex)
}
```

A outra barreira, um pouco mais discreta usa-se de uma estrutura própria da linguagem.

```
// Inicializacao
  sumPart = make(chan *big.Rat)

// Em cada processo de calculo do cosseno:
  sumPart <- sum
// significa insere o conteudo de "sum" no buffer de tamanho variavel sumPart.

// No processo original:
  for i := 0; i < q; i++ {
    cos.Add(cos, <-sumPart)
  }
// Significa para i=0 a q
    < await(buffer!=vazio);
    sum recebe proximo elemento do buffer >
    executa cos.Add(cos, sum)
```

Essa estrutura é chamada na linguagem de *canal*.
O trecho que está no processo original funciona como uma berreira, fazendo o processo esperar até que haja alguma mensagem no canal.

4 Biblioteca de alta precisão

Foi utilizado a biblioteca padrão da linguagem para número com alta precisão.

4.1 Naturais

Números naturais nessa biblioteca são representados como um vetor de tamanho variável de inteiros de 64bit sem sinal, onde 0's precedentes são eliminados a cada operação feita e, assim

sendo, o vetor vazio ou nulo representam o 0.

4.2 Inteiros

Inteiros são structs compostas por um natural e um booleano representando o sinal.

4.3 Racionais

Racionais são structs contendo um inteiro denominador b e um inteiro numerador a, sendo assim um racional na forma $\frac{a}{b}$. Sendo considerado o sinal apenas de a para determinar o sinal do racional. Sendo assim a precisão do número é tão grande quanto se desejar.

4.4 Vantagem

A grande vantagem do uso desta biblioteca é que é possível obter **qualquer** precisão, **não** apenas limitado aos **100k** exigidos pelo execício.

4.5 Desvantagem

A grande desvantagem do uso desta biblioteca é que não há controle algum sobre a precisão, o que torna as operações pesadas demais em pouquissimas rodadas do algorítmo, ou seja, ela é (muito) **lento**.