Diogo Pereira Lobo

Turma 1: Super Computação

Atividade 13 (código se encontra na pasta SCRATCH/aula13 do cluster Franky)

Schedulers

default: time: 0.000201159, time: 0.000228217, time: 0.000213511

schedule(static): time: 2.82815e-05, time: 3.27304e-05, time: 2.78624e-05

schedule(static, 4): time: 2.7434e-05, time: 3.16883e-05, time: 2.79173e-05

schedule(static, 8): time: 2.70838e-05, time: 3.60059e-05, time: 2.7366e-05

schedule(dynamic): time: 3.0973e-05, time: 3.50866e-05, time: 3.16333e-05

schedule(dynamic, 1): time: 2.95518e-05, time: 3.71123e-05, time: 3.66932e-05

schedule(dynamic, 4): time: 2.85264e-05, time: 3.26606e-05, time: 2.93115e-05

schedule(dynamic, 8): time: 2.8084e-05, time: 3.26093e-05, time: 2.89762e-05

schedule(guided): time: 2.99355e-05, time: 3.38256e-05, time: 2.99839e-05

schedule(guided, 2): time: 3.66522e-05, time: 3.2953e-05, time: 2.90088e-05

schedule(guided, 4): time: 2.76798e-05, time: 3.25078e-05, time: 2.83467e-05

schedule(guided, 8): time: 3.33879e-05, time: 3.25441e-05, time: 2.84556e-05

schedule(auto): time: 3.15579e-05, time: 3.08026e-05, time: 2.84594e-05

schedule(runtime): time: 3.38117e-05, time: 3.38862e-05, time: 3.13288e-05

Gráfico

Descrição gerada automaticamente com confiança média

Valores médios:

{'default': 0.00021429566666666665,

'schedule\_static': 2.9624766666666665e-05,

'schedule\_static\_4': 2.9013200000000002e-05,

'schedule\_static\_8': 3.01519e-05,

'schedule\_dynamic': 3.25643e-05,

'schedule\_dynamic\_1': 3.4452433333333336e-05,

'schedule\_dynamic\_4': 3.0166166666666666e-05,

'schedule\_dynamic\_8': 2.9889833333333336e-05,

'schedule\_guided': 3.124833333333333e-05,

'schedule\_guided\_2': 3.287133333333333e-05,

'schedule\_guided\_4': 2.9511433333333333e-05,

'schedule\_guided\_8': 3.146253333333333e-05,

'schedule\_auto': 3.0273299999999997e-05,

'schedule\_runtime': 3.30089e-05}

O scheduler que apresentou o menor tempo médio foi o **schedule\_static\_4**, com um tempo médio de **2.90132e-05**.

Nenhuma teve uma grande variação. As que apresentaram alguma mudança levemente significativa, foi de apenas alguns décimos de segundo.

Pode ter sido mero acaso ou a quantidades de dados para paralelizar, mas não houve uma diferença muito significativa entre os tempos de execução de cada schedule. Mas vale ressaltar que a dynamic e a guided, para alguns chunk sizes (número de iterações processadas em cada thread), tiveram valores de execução um pouco maiores, provavelmente pelo delay de esperar algumas threads terminarem o trabalho (dynamic) ou por algumas threads terem loop maiores de iteração no início (guided).

Ps: Não consegui entender por que a default teve um tempo tão diferente das outras, cerca de 10 vezes mais lento, já que, teoricamente, deveria ter o mesmo tempo da static padrão.

PI

cálculo do pi paralelo

for 2147483648 steps pi = 3.14159265358979 in 1.07949697133154 secs (MIN\_BLK 1024\*1024\*256)

for 2147483648 steps pi = 3.14159265358978 in 1.08087346423417 secs (MIN\_BLK 1024\*1024)

for 2147483648 steps pi = 3.14159265358996 in 4.51369129959494 secs (MIN\_BLK 1024)

cálculo do pi em tasks

for 2147483648 steps pi = 3.14159265358983 in 1.34980638604611 secs (num\_steps = 1024l\*1024\*1024\*2)

for 1048576 steps pi = 3.14159265358979 in 0.00188430026173592 secs (num\_steps = 1024l\*1024)

for 1024 steps pi = 3.14159273306265 in 0.000144052319228649 secs (num\_steps = 1024l)

O uso da diretiva task resultou em códigos mais rápidos, porém com valores mais distantes de pi a medida que o número de passos do cálculo "integral" diminui. Seria recomendado para tarefar cujo valor aproximado de pi, com poucas casas decimais, já é suficiente.

Já a diretiva parallel sozinha se mostrou eficiente e contante para o cálculo de pi (não mudei o valor dos passos) mas a alteração do MIN\_BLK teve impacto significativo no tempo de processamento, o que já era esperado, pois um MIN\_BLK pequeno pode resultar em uma quantidade excessiva de threads, resultando em overhead.

Claramente o uso de tasks com diferentes passos resultou em uma diferença mais significante, já que esse número está diretamente ligado ao cálculo do pi.

MANIPULAÇÃO DE EFEITOS COLATERAIS

com a diretriz critical

time: 0.00118872

time: 0.00132442

time: 0.00173004

com alocação de memória

time: 0.00047549

time: 0.000577642

time: 0.000505978

Fica claro aqui que a pré-alocação de memória se mostrou mais efetiva, pelo menos para pequenos valores, sendo cerca de 10 vezes mais rápida que o uso de região crítica.

Acredito que o uso de uma região crítica, fazendo cada thread rodar uma de cada vez, reduzindo o paralelismo, e para um cálculo tão simples, quanto multiplicar um valor por 2, culmina em uma alocação indevida de recursos de processamento, ou seja, em um overhead considerável.

Em ambas as abordagens o vetor é mantido na ordem. Na abordagem com região crítica, porque as threads respeitam o tempo de cada uma terminar para outra começar (sequencial) e na pre-alocação, mesmo que o vetor seja preenchido de maneira aleatória, a ordem dos índices se mantém.

Conclusões:

Em resumo, sempre se faz necessário avaliar bem o código, não apenas se é passível de paralelização, mas se esse processo resultará em melhorias para o código ou não.

Para problemas recursivos com efeitos colaterais, dependendo de como está o loop e o tamanho dos dados, usar pré alocação de memória se mostrou mais eficiente. Todavia, se possível, juntar os dois métodos vistos, seria mais eficiente, sõ não o foi para o caso analisado pois o loop de iteração no vetor, paralelizado ou não, teria uma execução sequencial.