Desafio de Programação Paralela

ERAD-SP 2018

Regras

Leia atentamente todos os problemas apresentados. Cada problema possui uma breve descrição, e exemplos de entradas e saídas (que podem estar nos anexos). Você pode criar sua própria versão paralela e/ou distribuída para o problema ou modificar uma implementação disponibilizada pelos juízes de acordo com sua estratégia, a não ser que o enunciado do problema diga o contrário.

Cada equipe deve criar um arquivo *Makefile* (veja os exemplos nos anexos) com as instruções de compilação. O arquivo binário principal gerado ou um script de execução para (e.g. MPI) deve ser igual a letra que representa o enunciado do problema, em maiúsculo: por exemplo, se o problema se chama ***A***, o executável/script deve-se chamar ***A***.

Para a submissão, cada equipe deve compactar o código fonte e o *Makefile* em um arquivo .zip (obrigatoriamente) e enviá-lo no sistema BOCA. Esse arquivo pode ter, no máximo, 32 Kb de tamanho. Submissões maiores do que este tamanho serão desconsideradas.

O tempo de execução de um problema será medido utilizando a ferramenta *time* no ambiente de testes dos juízes. Será considerada apenas o tempo real de CPU (*real CPU*). Cada submissão será executada três vezes com as mesmas entradas e o tempo médio será utilizado como a métrica de tempo de execução. O tempo de execução das soluções sequenciais dos problemas, produzido pelos juízes, também será medido da mesma forma. A pontuação se dará pelo *speedup*, ou seja, a razão entre o tempo de execução da solução sequencial pelo tempo de execução da submissão. As equipes ganharão pontos em cada problema conforme o valor do *speedup*, sendo este acrescentado no placar.

O time que tiver a maior pontuação será considerado a campeão do Desafio!

Problema A

Barra de Ferro

DARE é um jovem que gosta bastante de programar e ao mesmo tempo tem interesse nas mais diversas áreas do conhecimento. Ele realizou um experimento no qual colocava uma fonte de calor de um lado de uma barra (considerar sempre com 1 m de comprimento) que elevava a temperatura em 0,0001 ºC a cada segundo. Através de sua imaginação, ele bolou um método numérico para prever qual seria a temperatura na barra inteira a medida que o tempo fosse passando. Basicamente, seu método discretiza em n partes ($100 \leq n \leq 100.000$) a barra e calcula a nova temperatura pontualmente em cada intervalo, conforme o exemplo a seguir.

Barra dividida em 24 partes:

O método consiste na equação:

$$b_i = \frac{1}{2}(\alpha b_i + \beta b_{i+1})$$

onde b_i é a posição da i-ésima divisão da barra e α e β são fatores de correção no intervalo]0,1].

A extremidade da barra (local da fonte de calor) inicia com uma temperatura fixa T, informada pelo usuário e os demais pontos são considerados com valor ZERO.

Como DARE pretende testar seu método numérico diversas vezes, sabe que a versão serial é muito lenta e ainda não entende muito bem como paralelizar um código, ele pede que você utilize seus conhecimentos para elaborar uma implementação paralela de seu método a fim de conseguir diminuir o tempo total dos testes.

Entrada

A entrada é constituída de 5 valores: n, α, β, T e k, sabendo que o último parâmetro indica a quantidade de repetições do método, ou seja, o tempo de integração em

segundos ($10 \leq k \leq 2.592.000$).

A leitura dos dados é feita a partir da entrada padrão.

Saída

A saída é composta por *n* valores formatados com 2 casas decimais e representantes da barra seguidos de uma quebra de linha.

A escrita dos dados é feita na saída padrão.

Exemplo

Entrada
`10 0.9 0.92 10.0 100`
Saída respectiva a entrada
`2.00 2.40 2.87 3.43 4.10 4.90 5.86 7.00 8.37 10.01`

Problema B

Soma de Matrizes

Um dos problemas mais clássicos de programação, apresentados desde os primeiros cursos é a soma de matrizes bidimensionais. A representação de matrizes é usada para os mais diferentes fins, como a descrição de objetos na computação gráfica, a quantidade de determinado item armazenado em caixas com divisórias, entre muitos outros casos. DARE é um aluno aplicado de programação e já sabe como calcular a soma de matrizes bidimensionais. Porém, como ele não consegue se contentar com apenas o básico, ele decidiu alocar matrizes gigantes para fazer esse cálculo, mas percebeu que aumentando o tamanho das matrizes o tempo de processamento crescia também. Como ele adora desafios, mas não conhece muito bem as técnicas de paralelismo como CUDA, OpenMP ou MPI, ele pede que você o ajude a implementar essa soma, com alguma dessas técnicas, para que ele explore matrizes cada vez maiores. As matrizes têm tamanho n x m ($2 \leq n \leq 100.000$) e ($2 \leq m \leq 100.000$) e ambos devem ser considerados como pares.

Entrada

A entrada é constituída de 3 valores inteiros: n, m, e *seed*. O último é a semente para gerar números aleatórios para carregar as matrizes a serem somadas.
A leitura dos dados é feita a partir da entrada padrão.

Saída

Como escrever a saída demandaria um arquivo muito grande, DARE fez um cálculo de um número mágico para exibir na saída para validar a solução. A forma do cálculo não deve ser mudada. O formato de saída é um número real com 5 casas decimais.
A escrita dos dados é feita na saída padrão.

Exemplo

Entrada	Saída respectiva a entrada
`2 2 1`	`2.97500`