CAP-334 / 1997
Sexta Lista de Exercícios
Data de Entrega: 4/12/97, 10:00hs

Objetivo: O objetivo desta lista de exercícios é permitir a aplicação prática, através de exercícios de programação, dos tópicos vistos em aula sobre sistemas paralelos com memória distribuída e programação baseada em MPI.

Observações:

- Todos os programas desta lista devem ser executados, possivelmente de forma remota, a partir da estação `visap.nucleo.inpe.br`, uma Sparcstation do laboratório da CAP. Os códigos fonte necessários estão disponíveis na rede do laboratório, e podem ser acessados no diretório `~celso/lista6/`. Os programas devem ser compilados na máquina `visap.nucleo.inpe.br` (o ambiente de compilação de um programa com MPI está disponível apenas nesta máquina). Para poder executar os programas, deve ser criado também, na área de cada usuário, um arquivo chamado *.rhosts*, com uma linha tendo o seguinte conteúdo: `visap <username>`
- O diretório `~celso/lista6/` contém um Makefile que pode ser usado para compilar os programas. Basta fazer, por exemplo, "`make prog`". Há também um arquivo naquela área (`mpirun`) que permite a execução dos programas. Para executar, por exemplo, em quatro processadores, use "`mpirun -np 4 prog`".
- Em todos os exercícios, além de entregar uma cópia impressa com o código fonte e os resultados da execução dos programas solicitados, cada aluno deve também submeter uma cópia do código fonte via e-mail. Enviar cada programa separadamente, com o campo "Subject:" contendo a sigla "CAP-334(L-6)", para `celso@dpi.inpe.br`.

1) [20 pontos] Considere o programa `hello.c`, o qual imprime notificações informando o nome da máquina na qual cada processo estava sendo executado.

   (a) Compile este programa utilizando o Makefile indicado. Execute o programa com o comando "`mpirun -np 4 hello`", e apresente a saída gerada pelo programa durante a execução.

   (b) Modifique o código fonte do programa de tal maneira que as notificações geradas durante a execução estejam sempre em ordem estritamente *decrescente* de processadores. Compile e execute o novo programa usando os mesmos comandos acima. Apresente o conteúdo do novo código fonte, bem como a saída gerada pelo novo programa durante sua execução.

2) [20 pontos] Considere o programa `trap.c`, visto em aula, o qual aproxima o valor da integral de uma função `f(x)`. Modifique o código fonte deste programa de modo a calcular a integral da função $f(x)=3x^2+2x+1$ no intervalo [0,10]. Compile e execute este programa com **quatro** processadores. Apresente o novo código fonte, bem como o valor da integral de `f(x)` calculado na execução.

3) [30 pontos] O produto escalar entre dois vetores pode ser calculado, de forma paralela, através do programa `dot.c`. Os dados podem ser obtidos a partir de um arquivo de dados, fazendo-se redirecionamento do dispositivo de entrada, com a seguinte forma de execução: `"mpirun -np 4 dot <dados"`.

   (a) Compile e execute este programa com **dois** e com **quatro** processadores, utilizando os dados do arquivo `~celso/lista6/dados`. Apresente o valor final obtido para o produto escalar em cada um dos casos.

   (b) Modifique o código fonte deste programa, retirando a chamada à função `MPI_Allreduce()`, de modo a substituí-la por algum outro trecho de código que envolva *apenas* chamadas a `MPI_Send()` e `MPI_Recv()`. Compile e execute este novo programa com **quatro** processadores. Apresente o novo código fonte, bem como a saída gerada durante sua execução.

4) [60 pontos] De acordo com o que foi visto em aula, é comum modelar-se o tempo de transmissão de uma mensagem com n bytes através de uma função linear em n, do tipo $T(n) = A + B.n$, onde `A` é a latência e `1/B` a largura de faixa. Neste exercício devem ser construídos programas que permitam a determinação de `A` e `B` para as máquinas do laboratório da CAP, interligadas por rede Ethernet. Tais programas devem ser executados em **dois** processadores (`visap` e `lovelace`, duas Sparcstation-5). Um problema a ser contornado é o fato de que **não** existe um relógio global, comum aos dois processadores, para que se possa medir o tempo de *cada* mensagem. Assim, a solução normalmente adotada é criar duas mensagens em seqüência, uma de ida e outra de volta, dividindo-se pela metade o tempo total observado.

   (a) Construa um programa que transmita mensagens de `zero` bytes utilizando MPI, para determinar o parâmetro `A`. Durante a execução, a mensagem deve ser transmitida várias vezes, de modo a obter-se um valor estável para o tempo de transmissão. Apresente o código fonte deste programa, bem como o valor encontrado para a latência `A`, em segundos.

   (b) Modifique o programa anterior, de modo a poder enviar mensagens com tamanho variável de `zero` até `1MB`, com incrementos de `10KB`. O programa deve guardar em um array os diversos valores de tempo medidos (em segundos), e salvar tais valores num arquivo ao final da execução. Com estes valores, e utilizando um pacote matemático qualquer, determine a estimativa para o parâmetro `B`. Apresente este novo código fonte, bem como o valor obtido para a largura de faixa `1/B`, em Mbytes/segundo.