Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computacão e Automação

Programação Concorrente e Distribuida

Segunda Lista de Exercícios

Natal-RN, Brasil
[Outubro de 2017]

Professor

Prof. Samuel Xavier - DCA/UFRN

Aluno

Igor Macedo Silva - Bacharelando em Engenharia de Computação

Sumário

Lista de Figuras

Lista de Tabelas

1 Descrição

Lista da 2a unidade Descrição: Apresentar as respostas a todas as questões de exercício do livro texto, com exceção às questões: 3.3, 3.15 e 3.18.

Instruções para resolução (LEIAM!):

- Procure responder corretamente todas as questões da lista;
- Suas respostas serão validadas de forma oral por amostragem - geralmente de 2 à 3 defesas orais;
- Se não conseguir responder alguma questão, procure esclarecer as dúvidas em tempo em sala de aula com o professor, pelo SIGAA, com um colega, ou por e-mail. Se necessário, é possível marcar um horário para tirar dúvidas na sala do professor;
- Não serão aceitas respostas "mágicas", ou seja, quando a resposta está na lista entregue mas você não sabe explicar como chegou a ela. Sua nota nesse caso será 0 (zero). Mesmo que não saiba explicar apenas parte da sua resposta;
- Procure entregar a resolução da lista de forma organizada. Isso pode favorecer a sua nota;
- Os códigos dos programas requisitados (ou as partes relevantes) deverão aparecer no corpo da resolução da questão;
- A resolução da lista deverá ser entregue em formato PDF em apenas 1 (um) arquivo;
- O envio da resolução pode ser feito inúmeras vezes. Utilize-se disso para manter sempre uma versão atualizada das suas respostas e evite problemas com o envio próximo ao prazo de submissão devido a instabilidades no SIGAA;
- A lista com o número das questões respondidas deve aparecer na primeira folha da lista. Não será aceita alteração nessa lista.
- Procure preparar sua defesa oral para cada questão. Explicações diretas e sem arrodeios favorecerão a sua nota;
- A defesa deverá ser agendada com antecedência. Para isso, indique por email (samuel@dca.ufrn.br) no mínimo 3 horários dentro dos intervalos disponíveis em pelo menos 3 turnos diferentes. Caso não tenha disponibilidade em 3 turnos diferentes, deverá apresentar uma justificativa.
- Os horários disponíveis serão disponibilizados em uma notícia na turma virtual e serão atualizados a medida que os agendamentos forem sendo fixados.
- A defesa oral leva apenas de 10 a 15 minutos em horários fixados com antecedência. Não será tolerado que o aluno chegue atrasado para a sua prova.

Período: Inicia em 20/09/2017 às 00h00 e finaliza em 11/10/2017 às 23h59

2 Questões

2.1 Questão 3.1

What happens in the greetings program if, instead of `strlen(greeting) + 1`, we use `strlen(greeting)` for the length of the message being sent by processes `1, 2, ..., comm_sz- 1` ? What happens if we use `MAX_STRING` instead of `strlen(greeting) + 1` ? Can you explain these results?

Neste caso, o `+ 1` indica que o caractere de terminação da string também deve ser incluido no envio da mensagem. Se substituirmos por apenas `strlen(greeting)` a mensagem pode ser impressa corretamente ou não, dependendo do conteúdo presente no buffer de recebimento. Caso o buffer de recebimento esteja preenchido com zeros ("\0"), o comando `printf()` vai conseguir imprimir a mensagem corretamente mesmo que não exista um terminador nulo na mensagem enviada.

Em testes feitos localmente, as mensagens sempre foram exibidas corretamente, pois os buffers estavam sempre sendo iniciados com zero em suas posições de memória.

2.2 Questão 3.2

Modify the trapezoidal rule so that it will correctly estimate the integral even if `comm_sz` doesn't evenly divide n. (You can still assume that n $\geq$ `comm_sz`.)

Se `comm_sz` não divide perfeitamente n, devemos alocar os trapézios restantes nos processos de maneira mais deliberada. o pseudocódigo poderia ser:

```
1  get a, b, n;
2  h = (b - a)/n;
3  local_n = n/comm_sz; //Devemos garantir que a divisao sera inteira
4
5  n_mod_comm = n % comm_sz;
6  local_a = a + (my_rank*local_n* +
7                 my_rank*((int)(my_rank < n_mod_comm))+
8                 n_mod_comm*((int)(my_rank >= n_mod_comm && n_mod_comm
                      > 0)))*h;
9
10
11 local_b = local_a + (local_n + (int)(my_rank < n_mod_comm))*h;
12 local_integral = Trap(local_a, local_b, local_n, h);
```

O trecho da linha 7 se refere ao acrescimo incremental que deve acontecer ao h para cada rank. Isto é, no caso de `n_mod_comm = 3`, o primeiro `local_a` deve receber um acrescimo de 0, o segundo, de 1, o terceiro, de 3 e assim sucessivamente. Isso acontece pois é uma compensação ao `local_b` que está sendo acrescido de 1 até o momento em que todos os trapézios extras (no caso de n não exatamente divisivel por `comm_sz`) forem alocados em algum processo. E isso vai acontecer somente quando `my_rank` $\geq$ `n_mod_comm`

2.3 Questão 3.4

Modify the program that just prints a line of output from each process (`mpi_output.c`) so that the output is printed in process rank order: process 0s output first, then process 1s, and so on.

```c
#include <stdio.h>
#include <mpi.h>
#include <string.h> /* For strlen */

const int MAX_STRING = 100;
int main(void) {
    char phrase[MAX_STRING];
    int my_rank, comm_sz;

    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    if (my_rank != 0) {
        sprintf(phrase, "Proc %d of %d > Does anyone have a toothpick
            ?", my_rank, comm_sz);
        MPI_Send(phrase, strlen(phrase)+1, MPI_CHAR, 0, 0,
            MPI_COMM_WORLD);
    } else {
        printf("Proc %d of %d > Does anyone have a toothpick?\n",
            my_rank, comm_sz);
        for (int q = 1; q < comm_sz; q++) {
            MPI_Recv(phrase, MAX_STRING, MPI_CHAR, q, 0,
                MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("%s\n",phrase);
        }
    }

    MPI_Finalize();
    return 0;
}  /* main */
```

O princípio da resulução desta questão é perceber que para lidar com o não-determinismo do output de um programa MPI, nós devemos mandar todas as mensagens de saída para um único processo, neste caso o processo 0. Dessa forma, apenas um processo é responsável por gerenciar as mensagens de saída e, assim, podemos controlar a ordem de saída das mensagens.