Introdução à programação paralela

Gabriel Martins de Miranda – 13/0111350

1. Título do capítulo

Debugging your program.

2. Objetivo do capítulo

O capítulo é voltado a testar e debuggar um pequeno programa paralelo. A justificativa principal do capítulo é devido ao grande tempo que muitos programadores perdem apenas tentando encontrar ou remover erros no processo de desenvolvimento de software. O objetivo principal é mostrar formas de se desenvolver código gastando o menor tempo possível com depuração.

3. Resumo do capítulo

O processo de encontrar erros tanto para programas seriais quanto paralelos é cíclico, onde deve-se executar, encontrar erros e corrigir erros até que não sejam mais encontrados. Em programas paralelos, encontramos os erros comuns encontrados em programas seriais, além de problemas na comunicação, condições de corrida, starvation ou deadlocks. Em programas seriais, usamos primeiro dados simples como condição de teste e com o passar dos ciclos usamos dados mais complexos. Já no paralelos, é comum a cada ciclo aumentar o número de processos de execução.

Os três tipos mais comuns para encontrar erros são examinar o código fonte, adicionar saídas de erros e usar um programa à parte para encontrá-los. Com o uso da saída de erros, podemos captar estados das variáveis e execução do programa. Em sistemas paralelos isto pode mudar o comportamente do programa, situações como deadlocks e corridas podem aparecer ao usar a saída coletiva, sendo preferível cada processo escrever sua saída em arquivos diferentes.

Programas à parte, ou debuggadores, podem iniciar o programa, parar em vários pontos, examinar estados e fazer mudanças, entretanto este tipo de abordagem não é muito adequada para sistemas paralelos especificamente, sendo muito dependentes de sistema nestes casos.

Alguns erros comuns em programas paralelos são que dependem muito do sistema, apresentam deadlocks (quando todos os processos esperam por ações de outros e ficam bloqueados), recebimento ou envio de mensagens com argumentos errados, achar que são devido ao paralelismo mas serem serias, entre outros.

MPI provê algumas funções para gerenciar erros, sendo chamadas quando um erro ocorre. São locais aos processos e associadas a comunicadores. São pelo menos duas implementações, MPI_ERRORS_ARE_FATAL como o gerenciador padrão e MPI_ERRORS_RETURN para obter códigos de erros das funções do MPI. Para associar gerenciador de erros com um comunicador, é usado MPI_Errhandler_set e para encontrar significados dos erros, MPI_Error_string. Códigos de erro são dependentes da implementação, mas sempre serão menor que MPI_MAX_ERROR_STRING. Para abortar um programa após encontrar um erro, use MPI_Abort.

4. Solução dos exercícios

1- Debuggar o programa serial insertion sort

Ok. K estava a incrementar indefinidamente e era j>=0 como condição do laço for. Uma solução melhorada sem necessidade de k foi proposta.

2- Modificar comm_time.c para usar um comunicador com topologia unidimensional em forma de anel para passagem de mensagens

Ok. Modificado de comm_time.c versão 0.

3- Modifique comm_time.c de forma que overhead subtraído do tempo passado.

Ok. Modificado de comm_time.c versão 1.

6. Conclusão

Diversas técnicas para prevenir o aparecimento de erros na fase de codificação foram apresentados, além de técnicas para lidar com eles caso apareçam. Foram apresentadas as dificuldades encontradas para debuggar programas paralelos e a escassez de ferramentas para este tipo de implementação.

7. Referências consultadas

Parallel Programming with MPI by Peter Pacheco ; Debugging your program – chapter nine.