1º Trabalho

Processamento Paralelo e Distribuído 2016/01
Arquitetura Mestre-Escravo com Java RMI
Prof. João Paulo Andrade Almeida

27 de abril de 2016

O objetivo deste trabalho é praticar os conceitos básicos da programação paralela e distribuída. O trabalho deverá ser realizado em duplas. Plágio (cópia entre diferentes grupos) não será tolerado, e grupos com trabalhos copiados (mesmo que parcialmente) receberão nota mínima no trabalho como um todo. O trabalho deverá ser realizado usando middleware Java RMI.

A organização do seu código, clareza e detalhamento dos comentários é um critério importante de avaliação, assim como a qualidade do relatório.

Inclua os arquivos binários, o código fonte e o relatório (PDF) um arquivo `.zip` nomeado com `mestre-escravo` seguido dos nomes dos componentes do grupo separados por hífen. (Exemplo: `mestre-escravo-JoseSilva-AntonioVieira.zip`). Envie o arquivo para jpalmeida@inf.ufes.br com o nome do arquivo no campo "subject". Uma versão impressa do relatório deverá ser entregue na secreteria do Departamento de Informática na caixa postal/escaninho do professor. **O prazo para entrega é sexta-feira dia 27 de maio de 2016 (Não haverá prorrogação).**

1 Implementando arquitetura mestre-escravo

Implementar a arquitetura mestre/escravo como apresentada em sala de aula para realizar o cálculo de um "checksum" de um vetor de bytes de forma paralela. Um checksum (ou soma de verificação) é um código computado a partir de uma sequência de dados e que pode ser usado para verificar a integridade dos dados (ver https://pt.wikipedia.org/wiki/Soma_de_verificação).

Esta arquitetura consiste em um mestre e vários escravos. As interfaces do mestre e dos escravos devem oferecer uma operação de cálculo do checksum (de uma sequência de bytes). O checksum deve ser calculado de acordo com a abordagem de paridade com 8-bits (usando portanto um byte para o retorno). Nesta abordagem o checksum é o resultado da aplicação da operação ou-exclusivo (XOR) bit-a-bit na sequência de bytes de entrada.

Os escravos devem se registrar com o mestre ao serem inicializados, através de uma operação oferecida pelo mestre. Ao receber uma requisição do cliente, o mestre requisita os escravos registrados a computar em paralelo o cálculo do checksum de partes do vetor de entrada; após isso, o mestre produz o checksum do vetor de entrada completo a partir dos checksum parciais retornados pelos escravos.

O programa cliente deve gerar vetores aleatórios, de vários tamanhos, e imprimir os tempos de resposta da computação.

Use o serviço de nomes (RMI registry) para registrar e achar o mestre. Na presença de exceção na invocação de um escravo, o mestre deve remover este escravo da lista. Nos

escravos deve haver uma opção para terminar o escravo, que deve se desregistrar do mestre antes de terminar.

O sistema pode ser testado inicialmente em uma única máquina. (Claro, com vários escravos.)

Inclua no relatório todos os comandos utilizados para inicializar o registry, o cliente, o mestre e os escravos. Descreva que arquivos (.class) foram instalados em que máquinas.

2 **Análise do Desempenho**

Faça uma análise detalhada do tempo de resposta observado pelo cliente ao requisitar um cálculo. O objeto de estudo da análise é o *speed up* da solução paralela e distribuída. Inclua um relatório para essa análise, que deve considerar diferentes tamanhos do vetor, o tempo de resposta no caso sequencial e o caso de processamento paralelo e distribuído com vários escravos.

Crie gráficos do tempo de resposta para esta análise, usando (pelo menos):

- Diferentes tamanhos de vetor na faixa $[1..10^6]$

    (crie um gráfico do tempo de resposta x tamanho do vetor)

- Considere o caso sequencial (toda a computação feita pelo mesmo processo, sem uso de arquitetura cliente servidor), e o caso distribuído com pelo menos 2, 3 e 4 escravos (considere mais se possível) [atenção: cada escravo deve estar rodando em uma máquina diferente]

    (crie gráficos do tempo de resposta nesses casos x tamanho do vetor)

    (crie um gráfico com o *speed up* da solução com diferentes números de escravos x tamanho do vetor)

Inclua no relatório o código usado para estimar o tempo de processamento para o caso sequencial. Esta mesma implementação deve ser usada para implementar o cálculo do checksum dos subvetores nos escravos. (O caso sequencial deve ser implementado na mesma linguagem de programação para uma comparação justa.)

É recomendado que você crie um cliente especial para a avaliação, que automatize a avaliação, capturando muitas amostragens do tempo de resposta para os diferentes tamanhos de vetor. Dica: gere um arquivo com extensão .csv (http://en.wikipedia.org/wiki/Comma-separated_values) que pode ser aberto pelo Excel/OpenOffice para criação dos gráficos.

Documente: tipo de máquinas usadas, processadores, sistema operacional, memória disponível, configuração de rede. Se necessário, use o laboratório de graduação.

Estime o *overhead* de comunicação (a parte do tempo de resposta experienciado pelo cliente ao invocar a operação de cálculo do checksum e que é gasto com a transferências de dados entre o cliente e o mestre e entre o mestre e os escravos.) Inclua gráficos do *overhead* x tamanho do vetor.

O grau de detalhamento da análise é um fator importante para a avaliação (inclua outros gráficos e dados relevantes para a sua análise).

O que você pode concluir a partir da análise? Inclua suas conclusões no relatório.

Em uma máquina multi-core, faça também experimentos com uma solução potencialmente paralela porém não distribuída, ou seja com um escravo rodando para cada core na mesma máquina. Compare esta solução com a solução centralizada e com a solução distribuída.