PSPD – Programação para Sistemas Paralelos e Distribuídos
Turma A (Prof. Fernando W Cruz)

Lab02 – Construindo aplicações distribuídas usando *sockets* TCP/UDP

A) Objetivo: O objetivo desse experimento é que o aluno compreenda as características inerentes à construção de aplicações distribuídas, incluindo passagem de parâmetros, envolvendo módulos cliente e servidor usando *sockets* TCP/UDP.

B) Detalhes do laboratório

Versão com um 1 cliente e 1 *worker* (servidor) - No ambiente local (Linux), construir uma pequena aplicação distribuída (linguagem C), que descobre o maior e o menor valor em um vetor de 10.000 posições, considerando os seguintes passos:

1. O vetor deve ser gerado no cliente, deve ser de números do tipo *float*, e inicializado pelo uso da fórmula a seguir:

```
v[i] = (i - tamanho_do_vetor/2) ** 2;
```

2. Em seguida, a aplicação do lado cliente deve calcular cada posição do vetor, de acordo com a seguinte fórmula:

```
v[i] = raiz_quadrada(v[i]);
```

3. A aplicação deve ser organizada, de modo que a função que encontra o maior e o menor valor de um vetor esteja instanciada num servidor remoto (*worker*). O servidor (*worker*), portanto, deve analisar o vetor, encontrar o maior e o menor valor e repassar esses números de volta para o cliente, de modo que ele possa imprimir o resultado para o usuário.

4. Ou seja, o cliente apresenta para o usuário solicitante, o resultado da operação. Perceber que a solicitação foi realizada de maneira distribuída, ou seja, envolvendo unidades de processamento e espaços de memória distintos.

Versão envolvendo 1 cliente e múltiplos *workers* (servidores) - Nessa versão, a aplicação deve ser construída de modo que o cliente faça solicitação simultânea aos *workers*, visando melhoria de performance da aplicação.

C) Questões de ordem

- A atividade pode ser feita por grupos de até 2 alunos.
- Caso solicitados, os alunos devem estar preparados (slides e apresentação do código) para demonstração da solução em sala de aula, conforme definido pelo professor em data oportuna.
- A entrega será feita pelo envio de um arquivo zipado no ambiente Moodle da disciplina disponível em http://aprender3.unb.br. O arquivo zipado deve conter: *(i)* os arquivos .c das aplicações criadas separadas em pastas, uma para cada uma das versões comentadas, *(ii)* instruções de instalação e uso da aplicação criada, e *(iii)* um relatório cujo conteúdo está descrito mais adiante.

- Na versão com múltiplos *workers*, observar o seguinte:
    a. Se os códigos dessa solução forem diferentes em relação à solução com apenas um *worker*, os respectivos códigos devem ser incluídos na entrega. Nesse caso, é permitida a alteração completa do código, inclusive a lógica e os parâmetros passados entre os lados cliente e servidor, de modo a atender o problema.
    b. No código do cliente, ressaltar como é feita a coordenação dos trabalhos feitos por cada *worker* e como os resultados são consolidados e apresentados para o usuário (colocar comentários no código)
    c. Nessa solução, fazer testes com cenários envolvendo 2, 4, 6, 8 e 10 *workers*, de modo a verificar os valores de performance da aplicação em função da quantidade de processos envolvidos. Os tempos de resposta em cada uma desses cenários devem ser anotados e colocados em uma tabela, que fará parte do relatório desse lab.
- Os códigos de cada uma das versões descritas devem ser entregues, devidamente comentados e identados, e em pastas específicas.
- Deve ser entregue também um relatório (formato pdf) a ser entregue junto com o experimento, contendo os seguintes pontos:
    a) Título do experimento, dados da disciplina e identificação do(s) aluno(s) participantes
    b) Introdução – pequena descrição do problema e da arquitetura com *sockets*.
    c) Descrição da solução, mostrando a ideia da solução adotada, indicando qual dos dois tipos de *socket* - TCP ou UDP - foi utilizado para essa aplicação, com a devida justificativa (embasada por testes práticos).
    d) Descrição da solução com um *worker* – apontar problemas e soluções encontradas até chegar à versão final do problema proposto. Incluir aqui eventuais limitações dos códigos entregues.
    e) Descrição da solução com múltiplos *workers* - apontar problemas e soluções encontradas até chegar à versão final do problema proposto. Incluir aqui eventuais limitações dos códigos entregues. Importante:
        I. Descrever quais foram os mecanismos adotados no cliente de modo a coordenar os trabalhos dos *workers* e integrar os resultados numa solução única.
        II. Mostrar uma tabela apontando o desempenho da aplicação, em função da quantidade de *workers* testada.
    f) Conclusão sobre o experimento, ressaltando e justificando os valores encontrados nos cenários testados com múltiplos workers e principais diferenças de programação percebidas em relação à programação RPC. Além disso, cada aluno do grupo deve manifestar sua opinião sobre o laboratório, apontando eventuais aprendizados novos, as dificuldades encontradas e possíveis limitações percebidas no experimento. Deve ainda deixar claro como foi a sua participação no Laboratório (onde colaborou mais) e atribuir uma nota, no intervalo de zero a dez, em função da sua participação e aprendizado (lembrando que a nota da versão com vários *workers* tem peso maior do que a nota da versão com apenas 1 *worker*) .