Projeto de Implementação 1 – Sistemas Distribuídos

Objetivo do projeto: Trabalhar os conceitos de concorrência distribuída, exclusão mútua distribuída, consistência de dados distribuídos e tolerância a falhas por meio de um projeto de implementação em Java. Todas a comunicação (requisições e troca de dados de controle) entre os elementos do sistema deverá ser realizadas via sockets TCP.

Arquitetura do sistema: 1 cliente gerador de requisições, 1 balanceador de carga, e 3 servidores de aplicação multithread com 1 arquivo de dados local vinculado a cada servidor (totalizando 3 arquivos).

Cliente: Envia 2 tipos de requisição, escolhidas aleatoriamente (sorteio) para o Balanceador de Carga: Escrita e Leitura. Para requisições de escrita, devem ser sorteados dois números entre 2 e 1.000.000, que serão enviados a um Servidor de Aplicação (por meio do Balanceador de carga) para verificação do MDC (Máximo Divisor Comum) entre esses 2 números. O cliente deve salvar em um arquivo o log de todos os pares de valores enviados em requisições de escrita (1 linha para cada par de valores enviado). As requisições de leitura se destinam apenas a imprimir quantas linhas o arquivo de dados do Servidor de Aplicação que atender a respectiva requisição possui. Após um envio de requisição, o cliente dorme um tempo aleatório entre 20 e 50ms. Assim, os clientes ficam em loop infinito enviando requisições de leitura ou escrita, e dormindo.

Balanceador de carga: Distribuir as requisições de escrita aleatoriamente entre os 3 servidores de aplicação, ou seja, 1 requisição de escrita recebida do cliente será distribuída de forma aleatória (sorteio) para 1 servidor de aplicação. Já uma requisição de leitura será distribuída em broadcast para os 3 servidores de aplicação.

Servidores de Aplicação Multithread: Caso recebam uma requisição de escrita, devem fazer a verificação do MDC entre os dois números recebidos, escrevendo em uma linha do arquivo local a frase “O MDC entre X e Y é Z”, onde X e Y são os valores recebidos pelo cliente. Caso recebam uma requisição de leitura, eles devem imprimir na própria tela do servidor quantas linhas o seu respectivo arquivo possui. A verificação do MDC e contagem de linhas do arquivo devem ser realizados via threads. Ou seja, quando qq tipo de requisição for recebida, o servidor deverá instanciar sob demanda uma nova thread para atendê-la. Essa thread deverá realizar a sua tarefa, e ser finalizada ao término do processamento da requisição. Para que haja um cenário multithread, assim que uma requisição de escrita for recebida e a sua respectiva thread for instanciada (ou seja, antes de iniciar o processamento de fato da requisição de escrita), ela deverá dormir um tempo aleatório entre 100 e 200ms. Desta forma, toda thread de escrita terá o ciclo de vida *instanciada->dormindo->processando requisição ->finalizada.* Já a thread associada à requisição de leitura deve ser processada de imediato em sua totalidade, sem tempo de dormida.

Arquivos: Cada frase “O MDC entre X e Y é Z” deve ser uma linha do arquivo dos servidores de aplicação. Para que o estado de consistência entre os arquivos seja atendido, os 3 arquivos devem possuir o mesmo conteúdo ao final de uma requisição de escrita atendida por um deles, ou seja, todos os pares de valores gerados pelo cliente nas requisições de escrita devem constar nos arquivos locais dos servidores de aplicação, independente de qual servidor fez o atendimento original da requisição de escrita.

Restrições do sistema:

a) Uma operação de escrita só pode ser efetivada se as bases de dados estiverem consistentes entre si. Atentar para a necessidade de represamento de requisições dentro do sistema enquanto o estado de consistência entre os arquivos não for atingido. Ou seja, os Clientes não irão parar de enviar requisições, e vocês devem implementar algum mecanismo para que essas requisições sejam guardadas e atendidas posteriormente (na ordem que chegaram) quando o estado de consistência for retomado, ou seja, quando os 3 arquivos estiverem com o mesmo conteúdo. Já as operações de leitura não precisam aguardar o esquema de consistência, e podem ser atendidas de imediato nos Servidor de Aplicação.

b) Nenhuma requisição dos clientes pode ser perdida.

c) Cada servidor de aplicação escreve e lê (para contagem das linhas) apenas em seu arquivo local. Ou seja, o servidor 2 não escreve ou lê o arquivo do servidor 1, por exemplo. Desta forma, para escrever ou fazer a leitura em um arquivo, a operação deve obrigatoriamente passar pelo servidor de aplicação ao qual o arquivo está vinculado. Vocês devem levar isso em consideração no processo de consistência dos arquivos que vocês deverão implementar.

OBS: Vcs podem inserir livremente outros objetos destinados ao controle do sistema, utilizar elementos como buffers, arquivos auxiliares, etc., printar mensagens de controle ou informativa na tela, dentre outras ações, desde que essas ações não violem as regras de funcionamento e restrições do sistema.

Datas de entrega:

Até as 23:59 de 10/12: Entrega do código fonte na tarefa do teams

Aula de 11/12 (será iniciada às 8h no Lab de Programação): Cada dupla deverá explicar o seu código e executar a solução em sala (ordem por sorteio na hora). Cada dupla poderá levar um notebook ou providenciar um ambiente remoto (escolha livre) para execução do código entregue no Teams.

Pontuação do trabalho: 0-10 pontos, sendo 8 pontos para a corretude da solução e 2 pontos para o grau de eficiência da solução;

Nosso retorno às aulas presenciais será apenas no dia 17/12 (atentar para o ínicio as 8h), para que a carga horária da disciplina nessas 3 semanas possa ser dedicada à realização deste trabalho.

Qq dúvida é só chamar no Teams para esclarecimentos virtuais ou esclarecimento presencial no CEFET.

Bons estudos!