Trabalho: Replicação de processos e arquivos

Daniel S. Camargo – daniel@colmeia.udesc.br
Maurício A. Pillon – mauricio.pillon@udesc.br *(professor)*

16 de outubro de 2017

1 Introdução

A replicação é uma abordagem ampla, que permite fornecer alta disponibilidade de serviços e a tolerância à falhas em diversos contextos. Para que um projeto de software possa utilizar um mecanismo de replicação, sua arquitetura deve ser organizada por uma sequência de serviços, preferencialmente isolados e independentes entre si. Como exemplo prático, o atendimento de requisições é um serviço que deve estar separado do processo de busca em um banco de dados, isolado do próprio banco de dados, que constitui um terceiro serviço. Assim, desenvolvendo-os de modo independente, se permite replicá-los e configurar individualmente diferentes níveis de disponibilidade, escalabilidade e elasticidade.

2 Descrição

O trabalho a ser desenvolvido consiste em uma aplicação *multithreading* local que resolve o problema do empacotamento Bin Packing[1]. O programa deve fornecer duas soluções: força bruta e uma heurística de aproximação chamada *First Fit Decreasing* (FFD), utilizando-se de um espaço de memória compartilhado entre as Threads chamado de `deepCopy()` para realizar o armazenamento temporário da solução. Um código de base para o Bin Packing está disponível em: `https://github.com/DanielFloripa/BinPackingMultithread-SDI`, sendo que há um arquivo de entrada `input.txt` com três grupos de *itens* a serem alocados no *bins*.

A solução deve constituir três camadas de serviços básicos: *front-end* para ler os itens no arquivo texto de entrada, *back-end* para realizar o processamento dos itens

[1] Mais detalhes em: `https://en.wikipedia.org/wiki/Bin_packing_problem`.

requisitado e um serviço de monitoramento da saúde das threads. Para prover a alta disponibilidade, cada um dos serviços deve ser executado em n Threads em paralelo, com $1 \leq n \leq 4$ em cada teste (especificado abaixo), permitindo que estes serviços possam ser devidamente replicados. Para a tolerância à falhas, deve-se utilizar o conceito de *lockstep*, constituindo um serviço dedicado a verificar em cada passo de iteração, se a saída de todas as réplicas (respostas de cada thread no `deepCopy()`) estão consistentes antes de armazená-las. Devem ser executados testes no processamento por força bruta `BinPackingBruteforce.java` e no `FirstFitDecreasing.java`. Em resumo, o sistema a ser desenvolvido deve ter como serviços:

- Um *front-end* para atender as requisições dos clientes (leitura de uma linha do arquivo por vez, cada linha com n itens é uma requisição);
- Gerenciador de réplicas para todas as threads (de front e back-ends), com `health-check`;
- Serviço de *lockstep* para tolerância à falhas;
- Um serviço de *log* de sistema para salvar todos os estados do plano de testes, contendo dados sobre tempo, falhas causadas e tempo de troca de threads.

3 Plano de testes

O plano de testes especificado abaixo serve para guiar o aluno a monitorar a aplicação em relação ao seu desempenho. Exige-se um desempenho razoável, sendo a sua análise primordial para conhecer seu comportamento. Assim, deve-se verificar para cada etapa abaixo o tempo de recuperação de um determinado serviço quando na ocorrência das seguintes falhas: (i) kill no processo do front-end; (ii) kill no processo primário do back-end; (iii) alterar aleatoriamente o resultado de um passo de uma das Tthreads para ativar o *lockstep*. Para cada uma das requisições (linhas do arquivo), deve ser realizado os seguintes testes:

- Medição no início:
  1. Tempo inicial da requisição (t_0);
  2. Tempo final de uma resposta **sem** a ocorrência de falhas (t_{final} ótimo);
- Medição no BinPacking:

1. Aplicar Falhas (i), (ii) e (iii) e medir tempo de ação da troca de contexto e do *lockstep*, deve inclui as etapas de detecção e a troca de uma réplica para o primário)

- Tempo final de uma resposta **com** a ocorrência de falhas (t_{final} falhas);

Estes testes devem ser executados ao menos 10 vezes para cada uma das três falhas especificadas (indica-se que sejam automatizados, com tempos gravados em logs). Finalmente, os resultados estatísticos devem estar sumarizados em média, mediana e desvio padrão descritos em um relatório final formato SBC (com resumo). Este relatório deve ser curto (até três páginas), relatando além das estatísticas, as principais decisões de projeto que os alunos julgarem relevantes, citar o uso de alguma biblioteca, quando utilizada, e eventuais links de fontes usadas como base de desenvolvimento.

4 Considerações

O trabalho deve ser desenvolvido obrigatoriamente em duplas e em linguagem Java. Recomenda-se o uso de bibliotecas padrão do Java. Recomenda-se uma abordagem de automatização da execução e coleta dos tempos em logs. A apresentação será feita apenas para o professor Pillon e o monitor Daniel. Recomenda-se utilizar a mesma versão e arquitetura do Java instalado no laboratório para apresentação da execução. Em caso contrário, o aluno deverá trazer seu próprio notebook para demonstração, salientando que eventuais problemas de execução no dia da apresentação constitui em nota 0 no trabalho. São bem-vindos relatos sobre decisões de projeto/tratamentos adicionais para a abordagem de replicação, devendo ser reportados no relatório final.

Bibliografia

- Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2011). **Distributed Systems: Concepts and Design. International computer science series**. Pearson, Boston, 5 edition.
- Sobre o Bin Packing: `https://en.wikipedia.org/wiki/Bin_packing_problem`
- Código Base para o trabalho: `https://github.com/DanielFloripa/BinPackingMultithread-SDI`
- Treinamento sobre Java MultiThreading: `https://www.udemy.com/java-multithreading/learn/v4/overview`