Gerenciamento de Tarefas com ThreadPoolExecutor - Criando um Executor de Threads

Objetivo do Projeto

Este projeto demonstra o uso de um ThreadPoolExecutor em Java para gerenciar a execução concorrente de tarefas. Um servidor é configurado para receber e processar múltiplas tarefas utilizando um pool de threads, otimizando a execução e distribuição das cargas de trabalho.

Estrutura do Projeto

O projeto é composto por três classes principais:

Server: Gerencia um pool de threads e recebe tarefas para execução. Task: Representa uma tarefa que será executada em uma thread do pool. Main: Responsável por iniciar o servidor e submeter múltiplas tarefas para execução.

Funcionamento

O servidor cria um pool de threads com um número fixo de 10 threads. São submetidas 100 tarefas ao servidor. Cada tarefa simula uma execução com um tempo aleatório entre 0 e 10 segundos. O servidor monitora o número de threads ativas e o número de tarefas concluídas. O servidor é encerrado após todas as tarefas serem concluídas.

🗒️ Cenários teste

Foram realizados três cenários de testes para validar o comportamento do sistema:

1. Execução Normal

Configuração: Pool de 10 threads, 100 tarefas.

Resultado Esperado:

Tarefas distribuídas entre as threads do pool. O tamanho do pool e o número de threads ativas variam conforme a execução. Todas as tarefas são concluídas corretamente.

=======

2. Pool de Threads Pequeno

Configuração: Pool de 2 threads, 100 tarefas.

Resultado Esperado:

As tarefas são executadas em lotes de 2 por vez. O tempo total de execução aumenta devido ao menor número de threads. Nenhuma tarefa é perdida.

=======

3. Tarefas de Longa Duração

Configuração: Algumas tarefas com duração maior (até 15 segundos).

Resultado Esperado:

As tarefas de longa duração não bloqueiam a execução das demais. O pool gerencia a concorrência garantindo que as tarefas sejam concluídas corretamente.

🔧 Conclusão:

🔗

O uso do ThreadPoolExecutor demonstrou ser eficiente para gerenciar múltiplas tarefas concorrentes. A implementação permite a execução otimizada das tarefas, garantindo melhor uso dos recursos do sistema. O projeto pode ser expandido e aprimorado para atender a diferentes cenários de concorrência e escalabilidade.

💻 Desenvolvedor

🔗

Marcio Fonseca
👨‍💻

=======