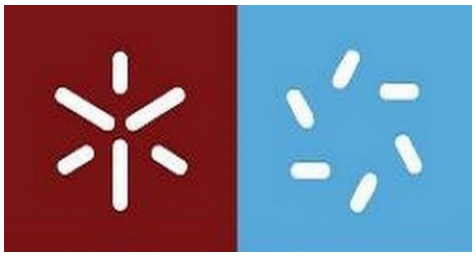

Universidade do Minho

Licenciatura em Ciências da Computação

Ano letivo 2023/2024

Sistemas Operativos – Trabalho Prático

Grupo nº18

João Sousa - A102462

João Matos - A102528

João Martins - A102496

CONTEÚDO

INTRODUÇÃO

É importante salientar que o projeto proposto acima tem como objetivo criar um serviço de orquestração de tarefas num ambiente computacional. Assim, os utilizadores poderão submeter as suas tarefas através de um programa cliente. Cada um deles fornecerá a duração prevista em milissegundos e a natureza das ações apropriadas. O sistema atribuirá a tarefa a um tempo designado, e um identificador único será atribuído à tarefa, enviando um feedback ao cliente. Por outro lado, o servidor tem a responsabilidade de escalonar e executar as tarefas dos utilizadores. Ao mesmo tempo, os dados produzidos por essas tarefas serão redirecionados para ficheiros utilizando o identificador da tarefa como nome. Isso facilitará o acesso dos utilizadores aos resultados.

Para além dos aspetos puramente técnicos, o projeto também está focado em fornecer a intuição e eficácia da experiência do utilizador. O programa cliente será criado para permitir a submissão de tarefas de forma fácil e fornecer informação em tempo real sobre o estado das mesmas. Além disso, os utilizadores poderão solicitar informações ao servidor sobre o progresso das suas tarefas e receber notificações sobre a sua conclusão. No servidor, serão implementados algoritmos de escalonamento inteligentes para distribuir as tarefas de forma justa, tendo em conta aspetos como a carga do sistema e os recursos disponíveis. Por fim, serão implementadas práticas de segurança para proteger os utilizadores e a informação armazenada.

A arquitetura do sistema será distribuída, utilizando tecnologias como computação em nuvem e contentores para garantir escalabilidade e fiabilidade. O objetivo é desenvolver uma solução robusta, eficiente e fácil de usar, que satisfaça as necessidades de processamento de tarefas num ambiente computacional dinâmico e em constante evolução.

ESTRUTURA DO PROJETO

O Orquestrador

No coração do sistema está o orquestrador, representado pelo arquivo “orchestrator.c”. Ele é o maestro responsável por aceitar solicitações de tarefas dos clientes e garantir que sejam executadas de maneira eficiente. Utilizando chamadas de sistema e estruturas de dados adequadas, ele gerência a execução das tarefas, mantendo uma lista organizada das solicitações e atualizando seu estado conforme necessário. O orquestrador também é encarregue de registar o andamento das tarefas num ficheiro de log, permitindo um monitoramento detalhado do sistema.

Os Clientes

Os clientes, representados pelo arquivo “client.c”, são os solicitantes de serviços no sistema de orquestração. Eles interagem com o orquestrador enviando solicitações de execução de tarefas ou requisitando informações sobre o estado das mesmas. Com base nas respostas do orquestrador, os clientes podem tomar decisões informadas sobre o prosseguimento das suas operações. Essa interação é mediada por mensagens, permitindo uma comunicação eficiente entre os diferentes componentes do sistema.

Definições e Estruturas de Dados

A estrutura do projeto é sustentada por um conjunto de definições e estruturas de dados, encapsuladas nos ficheiros “defs.c” e “defs.h”. Estes ficheiros estabelecem as bases necessárias para a comunicação entre o orquestrador e os clientes, definindo constantes, tipos de dados e declarações de funções essenciais para o funcionamento do sistema. Dessa forma, as operações são padronizadas e os componentes do sistema podem interagir de forma coesa e consistente.

Fluxo de Funcionamento

O fluxo de funcionamento do sistema é regido pela interação entre o orquestrador e os clientes. Quando um cliente envia uma solicitação, o orquestrador a recebe, processa e toma as medidas apropriadas para a execução da tarefa. Após a conclusão da tarefa, o orquestrador atualiza o seu estado e responde ao cliente, fornecendo informações relevantes. Esse ciclo de interação continua enquanto o sistema está em operação, garantindo a execução eficiente e coordenada das tarefas.

FUNCIONALIDADES DO CLIENT

1. Execute:

   O comando execute do cliente, é o comando mais importante deste programa pois é a partir dele que serão enviadas as tarefas ao orchestrator.
   As tarefas enviadas com o comando execute, podem ter flags, nomeadamente, “-p” ou “-u”, para caso sejam ou não, pipelines, embora a funcionalidade das pipelines não ficou completamente implementada, uma vez que, embora com uma função que nos permite a sua utilização, não conseguimos implementar a mesma neste programa.
   Neste programa, ao usar qualquer uma das flags, como as pipelines não foram implementadas, o programa apenas dará um write a avisar que a funcionalidade -p não se encontra implementada.
   No caso dos programas serão inseridos com a flag “-u”, estes correm da forma esperada, com os argumentos (provenientes do input dado pelo cliente. O output e o tempo de execução são registados num ficheiro criado após a primeira tarefa ser concluída, denominado de “task_log”, contendo o ID de cada tarefa e o tempo que a mesma demorou a ser concluída.

2. Status:

   Este comando é responsável por enviar um pedido ao estado das tarefas ao servidor. Os comandos que ainda se encontram em execução serão apresentados como “RUNNING”, tarefas em espera para execução estarão em “NEW” e as tarefas concluídas são denominadas de “DONE”

ORQUESTRADOR

O nosso orchestrator recebe apenas 2 argumentos, o ficheiro de output e as tarefas a serem executadas em paralelo, uma vez que apenas funciona com a política de escalonamento "FCFS". Foi decidido que a pasta do output, caso ainda não exista, não seja criada, por isso a pasta do output será a "tmp" por padrão.

No nosso programa "orchestrator", o tempo de execução das tarefas, é contabilizado a partir do momento em que a dada tarefa é enviada pelo "cliente" e é processada pelo "orchestrator", neste caso, o tempo apenas será usado para nos indicar quanto tempo cada tarefa demorou a ser executada.

O ficheiro "task_log", será criado a partir do momento em que a primeira tarefa for enviada pelo "cliente" e será usado para armazenar todas as tarefas que sejam concluídas, indicando o seu ID, os seus argumentos e o tempo que demorou a ser executada. Foi decidido, pelo nosso grupo, que tanto o "task_log" como os outputs das tarefas apenas serão eliminados ao utilizar o comando "make clean" para assim, os dados não se perderem de forma acidental. O output das tarefas, tal como o "task_log", serão criados após o envio de uma tarefa.

TESTES

Para testarmos as funcionalidades do programa, recorremos á utilização de scripts Bash que nos permitem testar os comandos, de forma que nos mostre casos inválidos e casos validos, o envio de várias tarefas em simultâneo e a mostrar o status das tarefas no fim de as enviar.

```
joao@LAPTOP-8076TE94:~/SO/grupo-18$ ./test_orchestrator.sh
Test 1: Sending execution request to orchestrator with invalid parameters
Usage: <output_folder> <parallel-tasks>
Test 2: Sending execution request to orchestrator with valid parameters
Infelizmente, o programa funciona apenas no modelo First Come, First Served (FCFS).
```

```
joao@LAPTOP-8076TE94:~/SO/grupo-18$ ./test_client.sh
Test 1: Sending execution request without enough arguments
Usage: ./bin/client execute time -u/-p "program [args]" OR ./bin/client status
Test 2: Sending execution request with invalid option
Invalid option
Test 3: Sending valid execution request
Task sent
Test 4: Sending status request
Status of tasks:

Tasks in execution:
Task 1: RUNNING

Scheduled tasks:

Completed tasks:
```

```
joao@LAPTOP-8076TE94:~/SO/grupo-18$ ./test_client2.sh
Test 1: Sending valid execution request multiple times
Task sent
Task sent
Task sent
Task sent
Task sent
Task sent
Task sent
Task sent
Task sent
Task sent
Test status: Sending status request
Status of tasks:

Tasks in execution:
Task 9: RUNNING
Task 10: RUNNING

Scheduled tasks:

Completed tasks:
Task 1: DONE in 2.04 ms
Task 2: DONE in 2.22 ms
Task 3: DONE in 2.03 ms
Task 4: DONE in 2.30 ms
Task 5: DONE in 2.40 ms
Task 6: DONE in 1.87 ms
Task 7: DONE in 2.82 ms
Task 8: DONE in 2.48 ms
```

Em termos de tempos, como só foi implementada a política de escalonamento FCFS, não é possível compará-la com as outras políticas com base em testes, apenas é possível observar que ao todo, para a realização das tarefas realizadas antes da execução do status, demoraram cerca de 18.16 milissegundos e demoraram em média, 2.27 ms.

CONCLUSÃO

Este projeto visa implementar um serviço de orquestração de tarefas em um ambiente computacional, utilizando comunicação entre processos e chamadas ao sistema em C. O sistema consiste em um cliente e um servidor que se comunicam através de pipes com nome (FIFOS).

O cliente permite aos usuários submeter tarefas para execução no servidor, fornecendo informações como o tempo estimado de execução e o programa a ser executado. O servidor é responsável por escalonar e executar essas tarefas, redirecionando a saída dos programas para arquivos com o identificador da tarefa. Além disso, o servidor regista em um arquivo o tempo de execução de cada tarefa finalizada.

Para melhorar a eficiência, o servidor utiliza a política de escalonamento First Come First Serve que executa as tarefas conforme a ordem de chegada. Os clientes podem consultar o estado das tarefas em execução, em espera e as terminadas, obtendo informações relevantes sobre os identificadores, programas e tempos de execução.

Em suma, este projeto proporciona uma oportunidade de aplicar conhecimentos práticos sobre chamadas ao sistema, comunicação entre processos e políticas de escalonamento em um contexto de desenvolvimento de software em linguagem C. Ele ilustra a importância da coordenação e do gerenciamento eficiente de recursos em ambientes computacionais para garantir um desempenho otimizado.