Universidade do Minho
Escola de Engenharia

Trabalho Prático

Sistemas Operativos

Grupo 69:

Duarte Faria nº 95609

Pedro Argainha nº 104351

Ricardo Jesus nº 100066

Introdução

Este relatório visa documentar o desenvolvimento e as funcionalidades de um sistema de orquestração de tarefas, implementado no contexto da disciplina de Sistemas Operativos da Universidade do Minho. Este sistema, concebido como um trabalho prático, destina-se à gestão eficiente de tarefas computacionais submetidas por múltiplos utilizadores através de um ambiente de rede simulado.

O objetivo principal deste sistema é facilitar a execução e o escalonamento de tarefas, permitindo que os utilizadores submetam tarefas especificando a duração esperada e o programa ser executado. Este relatório abrange desde a arquitetura e detalhes técnicos da implementação.

O programa foi desenvolvido utilizando a linguagem de programação C, com o sistema operativo Linux servindo tanto como ambiente de desenvolvimento quanto de execução. Este relatório detalha tanto o desenvolvimento do cliente, que interage com o utilizador via linha de comandos, quanto do orquestrador, que gere a execução das tarefas. A comunicação entre cliente e orquestrador é realizada através de pipes com nome, um método eficiente para a transferência de informações em sistemas baseados em UNIX.

Arquitetura

A arquitetura compreende dois componentes principais: o cliente e o orquestrador, que interagem entre si através de pipes com nome. Esta secção descreve a arquitetura detalhadamente, incluindo as especificações de ambos os componentes e a forma como eles se integram para fornecer a funcionalidade desejada.

1. Componente Cliente

O cliente opera através de uma interface de linha de comandos e é responsável por receber os comandos do utilizador, processá-los e comunicá-los ao orquestrador. O cliente permite ao utilizador especificar tarefas a serem executadas, fornecendo detalhes como o tempo estimado de execução e o programa a ser executado. A interface do cliente foi projetada para ser intuitiva, permitindo comandos como *execute* para iniciar tarefas e *status* para consultar o estado das tarefas.

Interface de comandos: A interface aceita comandos simples e em pipeline,

Comunicação com o servidor: o cliente comunica com o orchestrator através de um fifo denominado de fifoCliOrch para enviar o pedido ao orchestrator e o utiliza

o fifo com o id da tarefa que lhe foi atribuído para ler de volta a resposta do orchestrator.

Processamento de comandos: Descodifica e valida os comandos antes de enviá-los ao servidor.

2. Componente Servidor

O servidor é o núcleo do sistema de orquestração e é responsável por gerenciar todas as tarefas enviadas pelos clientes. Ele recebe comandos do cliente, executa as tarefas solicitadas, e mantém um registo das tarefas em execução, pendentes e completadas.

Gestão de tarefas: Recebe comandos de execução e status, e controla o ciclo de vida das tarefas.

Escalonamento: O servidor executa, num processo filho, o programa recebido do cliente assim que o mesmo chega ao orquestrador.

Execução de tarefas: Executa tarefas isoladamente, garantindo que a saída e erros de cada tarefa sejam redirecionados para arquivos específicos, com o nome do id da tarefa.

Persistência: Armazena informações sobre tarefas concluídas num ficheiro, permitindo consultas históricas e auditoria.

3. Comunicação

A comunicação entre o cliente e o servidor é realizada por meio de pipes com nome, uma escolha que suporta a comunicação em tempo real em um ambiente de múltiplos processos. Esta metodologia permite uma comunicação eficiente entre o cliente e o orquestrador. O orquestrador de cada vez que recebe dados do cliente, cria além de um processo filho para lidar com o pedido, um fifo com o id da tarefa atribuída ao pedido do cliente.

Implementação do orquestrador

O componente "Orquestrador" no sistema de orquestração de tarefas desempenha um papel crítico como servidor responsável por gerir todas os pedidos submetidos pelos clientes. Esta secção discutirá os principais aspectos da implementação do orquestrador, incluindo a sua estrutura e funcionalidades principais.

O orquestrador corre de forma continua e fica à espera de receber mensagens contidas numa estrutura package enviada pelo cliente.

Componentes chave:

Structs:

```c
typedef struct package {
    int id;
    char command[MAX_SIZE + 1];
    pid_t pid;
    int status;
    int command_type;
    int expected_time;
    long timestamp;
    struct package *next;
} Package;

typedef struct client {
    pid_t client_pid;
    Package *packages;
    struct client *next;
} Client;
```

Tanto o cliente como o orchestrator contém a estrutura package, contudo, o orchestrator contém a estrutura Client, utilizada para o armazenamento de packages recebidas dos clientes.

Loop principal: Um loop que processa as solicitações recebidas através dos pipes com nome.

Gestor de tarefas: Uma estrutura de dados que mantém o estado de todas as tarefas ativas, pendentes e concluídas.

Executor de tarefas: Um conjunto de operações que efetivamente inicia os processos correspondentes às tarefas em um ambiente isolado.

Funcionalidades Principais

O orquestrador possui várias funcionalidades projetadas para facilitar o gerenciamento eficiente de múltiplas tarefas simultâneas:

Receção e processamento de tarefas: O orquestrador aceita comandos para novas tarefas, extrai as informações necessárias e prepara as tarefas para execução. Ele gera um identificador único para cada tarefa, que é comunicado de volta ao cliente.

Execução e monitorização: quando é feito um pedido de execução de um programa pelo cliente, o orchestrator cria um processo filho, responsável pela execução do programa enquanto que o processo pai espera pelo fim da execução do processo e altera o estado do package para EXECUTED ou então para EXECUTED_ERROR em caso de erro.

Log e persistência: Registra informações sobre o início, término e duração de cada tarefa em arquivos para análise futura e auditoria.

Consulta de estado: Responde às solicitações de status dos clientes, fornecendo dados atualizados sobre tarefas em execução, pendentes e concluídas.

First-Come, First-Served (FCFS): Este algoritmo executa as tarefas na ordem em que são recebidas.

O orchestrator tem a capacidade de correr comando em pipeline.

Client

O componente "Cliente" no sistema de orquestração de tarefas é a interface direta com o utilizador, fornecendo um meio de submeter e gerenciar tarefas no servidor de orquestração. A implementação do cliente foca na eficiência na comunicação com o servidor. Esta secção aborda os principais aspetos da implementação do cliente, desde sua interface de linha de comandos até os detalhes técnicos de como os comandos são processados e comunicados ao servidor.

Componentes chave:

Struct package (como é possível observar na secção anterior): Lista ligada que contém o identificador , um array de caracteres com os comandos que o cliente pede ao orquestrador para executar, o tipo de comando (execute ou status), o tempo estimado de execução e timestamp. Nesta estrutura os campos id e time_stamp são inicializados a -1 para serem posteriormente alterados no lado do orchestrator. O campo command_type é inicializado com EXECUTE_COMMAND no caso de um

execute, EXECUTE_MULT_COMMANDS no caso de um comando em pipeline e com EXECUTE_STATUS no caso de um pedido de status. O campo status é inicializado sempre como NOT_EXECUTED.

Processador de comandos: Analisa e valida os comandos inseridos pelo utilizador antes de enviá-los ao servidor.

Submissão de tarefas: Permite ao utilizador submeter tarefas especificando o tempo estimado e o programa a ser executado. O cliente envia essas informações ao orquestrador e recebe um identificador único para cada tarefa.

Consulta de status: Oferece ao utilizador a capacidade de verificar o estado atual das tarefas, incluindo aquelas em execução, pendentes e concluídas.

Receção de respostas: Recebe e exibe as respostas do servidor, permitindo que os utilizadores recebam feedback imediato sobre as operações realizadas.

A implementação técnica do cliente envolve vários componentes críticos que garantem sua funcionalidade eficiente:

Análise de comandos: Utiliza um parser para decompor os comandos inseridos em componentes que o orquestrador possa entender. Isso inclui a separação de argumentos e a verificação da validade dos comandos.

Conclusão

O trabalho prático desenvolvido para a disciplina de Sistemas Operativos na Universidade do Minho apresentou um desafio significativo e uma oportunidade valiosa para aplicar conceitos teóricos em um cenário real de desenvolvimento de software. A implementação de um sistema de orquestração de tarefas em um ambiente Linux, utilizando a linguagem de programação C, exigiu um entendimento profundo dos mecanismos de sistemas operativos, bem como habilidades práticas em programação e design de sistemas.

A arquitetura dividida entre cliente e orquestrador provou ser eficaz para a gestão e escalonamento de múltiplas tarefas, infelizmente não foi implementado outros algoritmos de escalonamento para estudar a eficiência entre cada um dos mesmos. O cliente oferece aos utilizadores o papel de envio de tarefas, enquanto o orquestrador lida com o processamento e escalonamento dessas tarefas.

Em conclusão, este trabalho prático não cumpriu com os todos os objetivos académicos estabelecidos, principalmente a parte que o grupo acha mais interessante que é a implementação de diferentes algoritmos de escalonamento. O conhecimento adquirido e as soluções desenvolvidas neste contexto têm o potencial de serem aplicadas em outros projetos semelhantes ou expandidas em trabalhos

futuros para explorar novas funcionalidades e melhorias no sistema de orquestração de tarefas.