SISTEMAS OPERATIVOS

RELATÓRIO DO TRABALHO PRÁTICO

Simulador para *Offloading* de Tarefas no *Edge*

Diogo Miguel Henriques Correia
uc2016219825@student.uc.pt

Henrique José Gouveia Lobo
uc2020225959@student.uc.pt

Engenharia Informática
2.º Ano da Licenciatura

Faculdade de Ciências e Tecnologia da Universidade de Coimbra
2.º Semestre - 2021/2022

Descrição do funcionamento do programa

Quando o programa é iniciado, é feita a leitura do ficheiro de configuração e as informações são colocadas numa estrutura em memória partilhada, sendo que esta irá conter dados importantes para a execução do simulador. Abaixo encontra-se a descrição de diversos elementos presentes no programa:

- uma lista de servidores, com as capacidades dos seus vCPU's;
- flags, para saber quando ativar a *high performance* e quando terminar a simulação;
- uma matriz *servers_status*, que irá permitir saber, a cada momento, que servidor está disponível para receber tarefas para execução;
- uma *task_queue*, que irá servir para guardar as tarefas;
- duas variáveis de condição associadas à *thread dispatcher* e ao processo *monitor*;
- uma matriz *fd_pipe* que irá conter os unamed pipes associados a cada vCPU;
- duas matrizes de *mutexes* (*mutex_tasks* e *mutex_end_tasks)* que serão usadas na sincronização da execução das tarefas;
- uma matriz de threads chamada *vcpus* que será utilizada para simbolizar os vCPUS;
- uma matriz *tasks_times* que servirá de meio de comunicação entre os servidores e as threads vCPU's;
- um *mutex_log* para escrever no ficheiro de log;
- um *mutex_write* para aceder à memória partilhada.

Posteriormente, a thread scheduler vai receber tarefas ou comandos pelo named pipe. No caso de serem tarefas, estas são colocadas por ordem de prioridade na *task_queue*. Sempre que uma tarefa é colocada na queue, é feito um *signal* para uma variável de condição associada à thread dispatcher, para a avisar que pode começar a retirar tarefas da fila. Ao receber o comando "STATS", são impressas as estatísticas do simulador. No caso de receber o comando "EXIT" o programa coloca a *terminate_flag* com o valor true. Quando esta flag é ativada, a thread dispatcher irá enviar uma tarefa com um id especial para cada *edge_server* para estes saberem que têm de terminar. Por fim são impressas as estatísticas, os processos *edge_server*, *task_manager*, *monitor* e *maintenance_manager* são terminados e os recursos utilizados são removidos.

A *thread dispatcher* vai retirar a tarefa mais prioritária da *queue* e procurar um servidor que esteja disponível e que seja capaz de executar a tarefa em tempo útil. Se o tempo de execução da tarefa já tiver sido ultrapassado, ou se os servidores que estiverem disponíveis naquele momento não forem capazes de executar a tarefa em tempo útil, a tarefa é descartada. Caso contrário, é feito o cálculo do tempo que a tarefa irá demorar a ser executada e o resultado é colocado na posição correspondente da matriz *tasks_times.* Depois disto, a tarefa é enviada para o *edge_server* pelo unamed pipe a este associado.

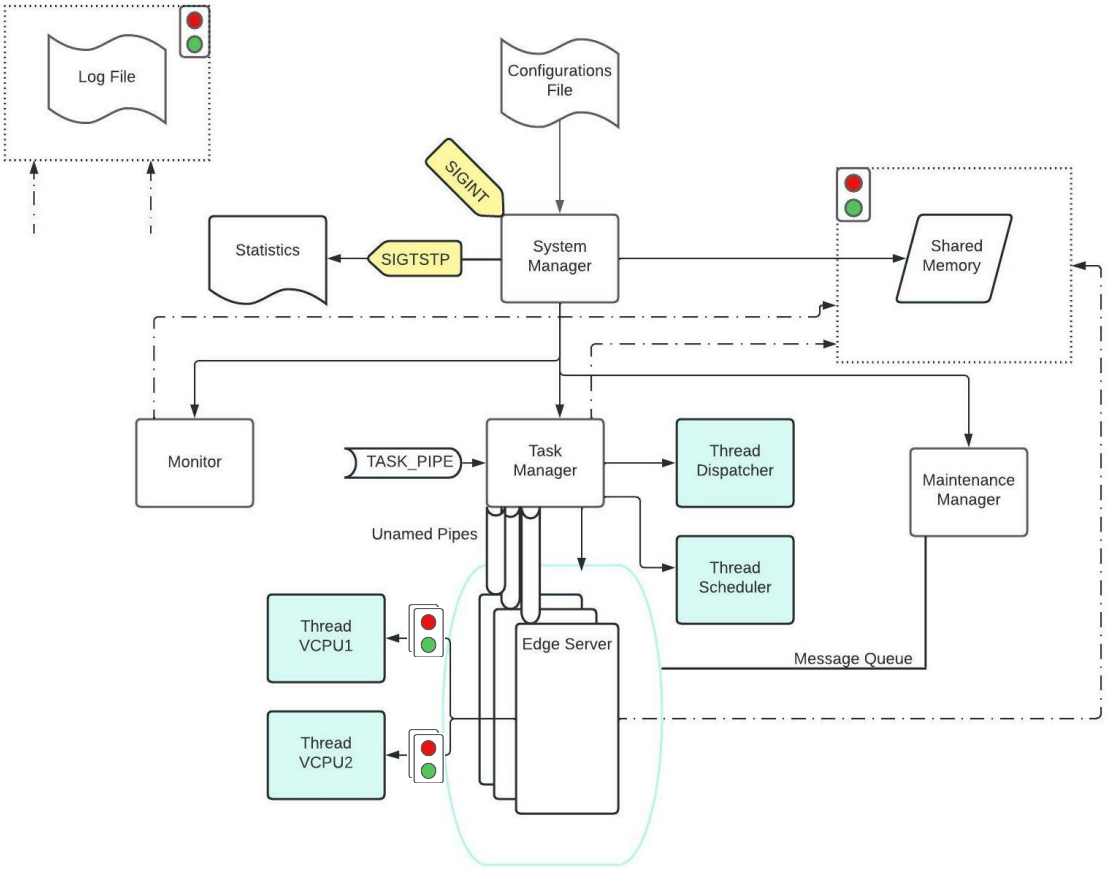
Quando o *edge_server* recebe a tarefa, é feito *unlock* ao *mutex_tasks* associado a cada vCPU de cada servidor e a *thread vCPU* irá fazer lock a esse semáforo e executar a tarefa. Após esta ter sido executada, é feito um *unlock* ao *mutex_end_tasks* associado e o *edge_server* irá fazer *lock* para reconhecer o fim da execução. De seguida, as variáveis de estatísticas são atualizadas, o status da vCPU é colocado a *AVAILABLE* e é feito um *signal* à variável de condição associada à thread dispatcher.

Foi também implementada uma versão mais simples do processo *maintenance_manager*. Este irá, de tempo a tempo, enviar mensagens para a *message_queue* com um id associado a cada *edge_server*. Quando os servidores terminam de executar uma tarefa, vão verificar se têm alguma mensagem para eles e, caso tenham, ficam em *sleep* durante o tempo estipulado pelo *maintenance_manager*.

No caso do processo *monitor*, este foi implementado, mas não está funcional. A thread a ele associada é "acordada" sempre que uma tarefa é adicionada ou retirada da *task_queue* e este vai fazer a verificação para aumentar ou diminuir a performance dos servidores.

Relativamente à receção e tratamento de sinais, foi apenas implementado o caso do SIGTSTP que, ao ser recebido, é feita a impressão das estatísticas. O tratamento do SIGINT não foi implementado, pois surgiram dificuldades ao fazer *wait* aos diversos processos.

Arquitetura do Programa



Horas de Trabalho

Diogo Correia: 30 horas
Henrique Lobo: 6 horas