Projeto Final de Sistemas Operativos
Simulador para Offloading de Tarefas

Ricardo Guegan 2020211358
Samuel Machado 2020219391
PL4

Abordagem:

Ao ler o enunciado e ao perceber a função dos diferentes processos, notámos que estes estariam constantemente em concorrência pelo o direito de escrita no ficheiro de log, na struct comum de stats e nas structs de dados acerca de cada edge server.

Outros problemas seriam o evitar do busy-waiting e a sincronização dos vários processos, visto que estes se encontram interligados e nem sempre podem executar a sua função num determinado momento.

Problemas de partilha de recursos:

No que toca à partilha dos recursos e maximização da concorrência, decidimos que, por cada edge-server teríamos uma struct com os seus dados. Assim, ao precisar de ler ou escrever dados apenas referentes a este, podemos concorrer apenas com este. Esta struct é regulada por um mutex presente na mesma.

No que diz respeito ao ficheiro log e à struct comum, estes são os maiores pontos de concorrência e, na nossa abordagem, incontornáveis, sendo eles regulados por um mutex.

Problamas de Sincronização:

Como quase todos estes dependem de outro para poder realizar a sua tarefa, é importante que estes consigam esperar em repouso. Os principais problemas de sincronização encontram-se no Task Manager, onde é necessário sincronizar as threads schedueler e task manager (este último recebe as tarefas) de modo a que, quando se receba uma tarefa, o schedueler a coloque e, finalmente, o dispatcher a entregue. Entre o criar e o escalonar utilizamos uma variável de condição sobre uma tarefa. Quando esta existe, é escalonada, e, quando não, continuamos a aguardar a sua eventual chegada ao pipe. O outro grande problema está presente entre o dispatcher e os vcpus, visto que um vcpu apenas está ativo enquanto realiza a tarefa, o dispatcher apenas está ativo enquanto existem vcpus livres e tarefas por distribuir.

Para conseguir sincronizar o dispatcher, utilizamos dois semáforos, sendo que um contabiliza o número de vcpus livres e, o outro, o número de vcpus ocupados. Já quanto aos vcpus, utilizamos variáveis de condição na qual a condição de ativação é a existência de uma tarefa que o tem como destino.

Gestão de tarefas:

Partindo do objetivo de maximizar o número de tarefas realizadas em tempo útil, decidimos que o nosso escalonador utilizaria um algoritmo SPN. Foi ainda necessário ter em atenção a regular reavaliação da validade de cada tarefa em fila de espera, bem como a verificação da sua exequibilidade, tendo em conta a disponibilidade e potência dos vcpus.

Horas de trabalho:

Acreditamos que o tempo necessário para realizar o projeto em questão deve estar próximo das 35h.

Nota: o diagrama seguinte encontra-se disponível em PDF no ficheiro .zip submetido.