4
Modelo Proposto

A análise dos trabalhos apresentados no capítulo anterior evidencia a convergência na utilização de quantidades reduzidas de *kernel threads*, o que denota a importância dessa característica para a estruturação de soluções para programação concorrente superiores ao *multithreading*. A utilização comedida de *kernel threads* e a exploração de programação concorrente com *user threads* não se esgota apenas nos trabalhos apresentados no capítulo anterior e pode ser observada também em outros trabalhos relacionados (28).

O interesse por novos modelos para programação concorrente dissociados do uso exclusivo de *kernel threads* e que permitam a exploração de *user threads* foi um dos principais motivadores do desenvolvimento do modelo para programação concorrente em Lua, objeto deste trabalho e descrito neste capítulo.

Nesse sentido, o modelo proposto adota a utilização comedida de *kernel threads*, em prol de *user threads*, para permitir a exploração de *multithreading* com menos restrições de escalabilidade. As *user threads*, que nesse caso são compostas por código Lua, são denominadas *processos Lua*.

Os processos Lua representam fluxos de execução independentes de código Lua e são entidades escalonadas exclusivamente por um escalonador executado no espaço no usuário, sem relação direta com processos do sistema operacional ou quaisquer entidades escalonadas diretamente pelo *kernel*. A dissociação entre os processos Lua e as entidades escalonadas diretamente pelo *kernel* implica escalonamento MxN.

A adoção de uma arquitetura híbrida, composta por *kernel threads* (em menor escala) e por *user threads*, permite ao modelo proposto agregar as principais vantagens e contornar as principais desvantagens de cada um desses modelos de programação. Enquanto a utilização de *kernel threads* viabiliza a exploração de paralelismo em ambientes multiprocessados, o que não seria possível apenas com *user threads*, a utilização de *user threads* viabiliza a criação e gestão de grandes quantidades de fluxos de execução simultâneos, o que não seria possível apenas com *kernel threads*.

A comunicação entre os processos Lua é realizada exclusivamente atra-

vés de troca de mensagens. O modelo de comunicação por troca de mensagens é baseado na utilização de primitivas para o envio de mensagens e para o recebimento de mensagens. Normalmente é possível utilizá-lo tanto para a comunicação, como para a sincronização de processos. Ainda que potencialmente mais lenta quando comparada com modelos de comunicação por memória compartilhada, a troca de mensagens é mais flexível, pois se adapta bem tanto à comunicação entre processos em uma mesma máquina quanto à comunicação de processos distribuídos em máquinas interligadas por redes.

A ausência de compartilhamento de memória entre processos Lua contribui para reduzir a complexidade no desenvolvimento, em particular por tornar desnecessária a utilização de mecanismos de sincronização baseados em memória compartilhada, o que endereça uma das principais críticas ao *multithreading* preemptivo com memória compartilhada. Adicionalmente, a execução do código de cada processo Lua em seu respectivo estado facilita a contenção de falhas.

As mensagens são representadas por cadeias de caracteres (*strings*). É possível transmitir mais de uma cadeia de caracteres com uma única chamada à função de envio de mensagens. O envio de dados diferentes de cadeias de caracteres ou números, que são convertidos em cadeias de caracteres, resulta na transmissão de valor nulo. No entanto, a restrição de envio de cadeias de caracteres e números pode ser facilmente contornada mediante a utilização de mensagens compostas por código Lua. Assim, nada impede que o conteúdo de uma mensagem seja composto por cadeias de caracteres que representem o código de uma função Lua cujo retorno inclua dados de qualquer tipo.

O endereçamento das mensagens é especificado por canais. Um canal é uma entidade própria, externa aos processos Lua, designada por um nome atribuído no momento da criação. O envio e o recebimento de mensagens requerem a especificação de um canal para o qual a mensagem será enviada ou a partir do qual a mensagem será recebida, respectivamente.

Chamadas à função de recebimento de mensagens resultam no recebimento da mensagem mais antiga enviada para o canal, ou seja, como em uma fila, as mensagens são entregues na ordem em que foram recebidas pelo canal. Não há garantias com relação à ordem na qual mensagens enviadas por processos Lua distintos para um mesmo canal são recebidas pelo canal. Cada mensagem só pode ser recebida uma vez por um único processo Lua, ou seja, o recebimento representa o consumo da mensagem.

O envio de mensagens é sempre síncrono. Chamadas à função de envio somente retornam o controle ao código Lua quando a mensagem é recebida por outro processo Lua ou quando o canal de comunicação correspondente é

destruído. Dessa forma, ao tentar enviar uma mensagem, um processo Lua deve permanecer bloqueado até que ocorra algum desses eventos.

O recebimento de mensagens pode ser síncrono ou assíncrono, de acordo com os parâmetros utilizados na chamada à função. O recebimento síncrono funciona de maneira análoga ao envio síncrono: chamadas à função de recebimento somente retornam o controle ao código Lua quando a mensagem é enviada por outro processo Lua ou quando o canal de comunicação correspondente é destruído. O recebimento assíncrono, por outro lado, retorna o controle ao código Lua imediatamente após verificar se há envio de mensagem pendente no canal. Em caso afirmativo, a mensagem é recebida. Em caso negativo, é retornado um erro indicativo da ausência de mensagens disponíveis para recebimento.

Ainda que a função de envio de mensagens seja síncrona, a biblioteca não exclui a possibilidade de implementação de comunicação assíncrona. Uma forma de fazê-lo consiste em disparar novos processos Lua cujo único propósito seja executar chamadas à função de envio de mensagens. Como a criação de novos processos Lua é assíncrona, o controle é imediatamente retornado ao código Lua original e eventuais bloqueios somente atingiriam os novos processos criados para realizar a comunicação.