7
Conclusão

Este trabalho apresentou um modelo alternativo para programação concorrente em Lua. Esse modelo é caracterizado pela utilização de processos Lua, *user threads* representadas por estados Lua, e troca de mensagens, em detrimento de compartilhamento de memória, como única forma de comunicação entre fluxos de execução. Os bloqueios são associados às primitivas de envio e recebimento de mensagens e conseqüentemente é possível prever onde e quando podem ocorrer.

A ausência de compartilhamento de memória elimina a necessidade de mecanismos para controlar o acesso a variáveis compartilhadas e regiões críticas, o que simplifica o processo de desenvolvimento e reduz a probabilidade de inconsistências que podem resultar em dados corrompidos ou anomalias durante a execução. A previsibilidade dos bloqueios, por sua vez, facilita a depuração e aumenta o determinismo do fluxo de execução.

A linguagem de programação Lua, ainda que não tenha sido desenvolvida especificamente para programação concorrente, mostrou-se flexível o suficiente para permitir a implementação, a contento, do modelo proposto. Além disso, proporcionou bom desempenho e boa escalabilidade à biblioteca desenvolvida, como evidenciado pelos resultados da avaliação de desempenho.

A utilização da biblioteca Native POSIX Threads Library (NPTL), mesmo que comedida, como fundamento para interação com *kernel threads*, permitiu a exploração do paralelismo intermediado pelo sistema operacional. No entanto, paradoxalmente, resultou também no aumento da complexidade do processo de desenvolvimento e na necessidade de lidar com vários dos obstáculos impostos pela utilização de *multithreading* preemptivo com memória compartilhada.

As dificuldades enfrentadas durante o desenvolvimento da biblioteca apenas reforçaram a procedência das críticas ao *multithreading* e evidenciaram a necessidade de novos modelos para programação concorrente. As limitações do *multithreading*, em particular a complexidade no desenvolvimento, dificultam até mesmo a sua utilização apenas como base para estruturação de alternativas superiores para programação concorrente.

A elaboração deste trabalho não esgota a investigação do modelo proposto para programação concorrente em Lua, tampouco a exploração de novas alternativas para programação concorrente. A biblioteca desenvolvida poderia ser incrementada por meio da implementação de novas funcionalidades, tais como: a definição, em tempo de execução, de limites de tempo para o recebimento de mensagens e o envio de mensagens para mais de um canal de uma só vez, ou seja, com uma única operação de envio (*multicast*). Adicionalmente, seria interessante avaliar a utilização da biblioteca para o desenvolvimento de aplicações mais complexas ou de maior porte, onde os potenciais de usabilidade e desempenho pudessem ser mais explorados.

Outra possibilidade seria a integração (ou atuação conjunta) com o projeto ALua (1), com o objetivo de fortalecer o potencial de utilização do modelo proposto em ambientes distribuídos. Em teoria, seria possível aliar as funcionalidades providas pelo projeto ALua para execução em ambientes distribuídos com as funcionalidades providas pela biblioteca desenvolvida para exploração do paralelismo em microcomputadores multiprocessados, o que resultaria em combinação ótima para exploração de ambos os recursos em todo o seu potencial.

A avaliação comparativa com Erlang, uma linguagem de programação desenvolvida para programação concorrente e amplamente reconhecida por sua escalabilidade, foi superficial e análises mais detalhadas da implementação ou utilização da linguagem não foram contempladas pelo escopo do trabalho. Entretanto, seria interessante avaliar algumas características adicionais, como por exemplo o consumo de memória ocasionado pela transmissão de mensagens, bem como a flexibilidade da linguagem e a dificuldade (ou facilidade) de desenvolvimento.

Outra importante e freqüentemente exaltada característica de Erlang, que não foi contemplada pelo trabalho em questão, é a tolerância a falhas (26). Na prática, a linguagem conta com uma série de funcionalidades voltadas a identificar, reportar e reagir a falhas em processos. A investigação de funcionalidades desse tipo e sua possível incorporação ao modelo proposto e à biblioteca desenvolvida pode ser vantajosa, em particular no caso de execução em ambientes distribuídos.

Finalmente, a disponibilização pública do código-fonte da biblioteca desenvolvida permitirá que novos esforços sejam empreendidos no sentido de divulgar, manter e expandir as duas principais contribuições deste trabalho: o modelo alternativo para programação concorrente em Lua e a própria biblioteca.