1
Introdução

Até o passado recente, a evolução dos processadores foi direcionada principalmente para o aumento da velocidade do *clock*, a otimização da execução de instruções e o aumento do *cache* acoplado (29). Avanços em qualquer um desses aspectos, em particular na velocidade do *clock*, favorecem o aumento de desempenho de qualquer software.

No entanto, em determinado momento, questões físicas associadas à dissipação de calor e ao consumo de energia, entre outras, começaram a representar obstáculos para o aprimoramento dos processadores. A taxa de aumento da velocidade do *clock* dos processadores, em especial, foi reduzida drasticamente e é possível observar como as velocidades permaneceram praticamente constantes nos últimos anos, apesar do lançamento de novos processadores.

O surgimento desses obstáculos suscitou uma mudança fundamental na evolução dos processadores, que passou a ser orientada principalmente à exploração do potencial de paralelização de execução de instruções. Indício inicial dessa mudança foi o desenvolvimento da tecnologia *hyperthreading*, que permite a execução paralela de algumas instruções em um mesmo processador. Entretanto, a vertente mais popular e atual da exploração do paralelismo está nos processadores multinúcleo, que compreendem em um único chip dois ou mais processadores, dotados de *caches* individuais, capazes de executar fluxos distintos em paralelo.

Uma das principais conseqüências dessa mudança está no fato de que modernizar o processador não mais proporciona, necessariamente, aumento de desempenho de software. Como a exploração do paralelismo para aumentar o desempenho não depende apenas do hardware, é preciso que o software seja desenvolvido de tal forma que utilize criteriosamente esse recurso. Dessa forma, a programação concorrente é apontada como instrumento essencial e progressivamente necessário à utilização plena dos processadores e à otimização do desempenho de software.

Apesar de sua importância, a programação concorrente ainda é predominantemente baseada em modelos e padrões ultrapassados, alvos de críticas recorrentes e notoriamente complexos. Além disso, o suporte robusto à con-

corrência em linguagens de programação e bibliotecas de software amplamente utilizadas ainda é parco (30). Esse cenário estimula a reavaliação dos modelos e padrões atualmente utilizados, bem como a proposição de alternativas superiores para programação concorrente.

O objetivo deste trabalho é explorar recursos de Lua (22), uma linguagem interpretada, com tipagem dinâmica e gerenciamento automático de memória, para estruturar e avaliar um modelo alternativo para programação concorrente. O capítulo 2 apresenta uma análise crítica do *multithreading* preemptivo com memória compartilhada, um modelo amplamente utilizado para programação concorrente. O capítulo 3 descreve sucintamente alguns trabalhos que apresentam propostas alternativas para programação concorrente. O capítulo 4 apresenta o modelo alternativo proposto para programação concorrente em Lua e detalha as suas principais características. O capítulo 5 descreve uma biblioteca desenvolvida como parte deste trabalho para implementar o modelo proposto. O capítulo 6 apresenta os resultados da avaliação dessa biblioteca e de comparativos com alguns trabalhos relacionados. Finalmente, o capítulo 7 apresenta algumas conclusões e sugestões de trabalhos futuros.