

Scheduling di programmi paralleli su multiprocessore

Algoritmo **OBLIVIO**: MFQ su multiprocessore come per programmi sequenziali. Questo però crea problemi con molti pattern comuni di parallelizzazione:

bulk synchronous programming ciclo

calcolo → barriera → comunicazione → barriera

se un singolo thread viene interrotto dallo scheduler, tutti vanno in stallo;

produttore-consumatore / pipeline interrompere uno stadio sospende l'intera catena;

cammino critico in generale, è la sequenza minima di passi che un programma parallelo deve compiere per andare a termine. Rallentare l'esecuzione del cammino critico impedisce l'avanzamento di tutti gli altri thread;

spin-then-wait ottimizzazione per cui i lock fanno busy waiting per un breve periodo di tempo prima di sospendere il thread. Se il thread che detiene il lock viene deschedulato, questa strategia è controproducente.

Queste situazioni non si presentano se i thread di un programma vengono sempre eseguiti simultaneamente (*gang scheduling*). Funziona per macchine dedicate prevalentemente all'esecuzione di un programma, ma non è conveniente altrimenti. Un'alternativa è assegnare un sottoinsieme dei core a ciascun programma e fare gang scheduling su quel sottoinsieme (*space sharing*).