MODULO A

Architetture moderne, codice seriale e memoria condivisa (40h)

Prerequisiti

- Familiarità con l'ambiente Linux e l'uso della riga di comando (shell, editor, compilazione, gestione dei file).
- Familiarità con il linguaggio C/C++ (compilazione, puntatori, gestione della memoria, strutture dati di base).
- È utile, ma non indispensabile, una conoscenza di base di analisi numerica e algebra lineare.

Contenuti didattici

- Architetture moderne (CPU): gerarchia di memoria e cache, branching e branch prediction, instruction-level parallelism — esecuzione out-of-order, super-scalarità, pipelining.
- Ottimizzazione del codice seriale: scrittura di codice efficiente su CPU moderne; vettorizzazione/SIMD (report del compilatore, dipendenze nei loop, gather/scatter, masking, allineamento); località dei dati e ottimizzazione della cache.
- Architettura del nodo e NUMA: organizzazione del singolo nodo, effetti NUMA su accesso e allocazione della memoria.
- Multithreading con OpenMP: dalle basi (parallel regions, work-sharing, scheduling) al task parallelism; OpenMP memory model e sincronizzazione tra thread (barriere, critical/atomic, flush).
- Debugging & Profiling I (seriale): flag sanitizer di gcc, gdb, profiling basato su PMU (perf, PAPI), valgrind.

Durata

40h — CPU+serial opt ~16h · NUMA+OpenMP ~16h · Debug/Profiling I ~8h

Obiettivi formativi specifici

Comprendere come le scelte di codice si traducano in performance sulla microarchitettura; saper portare un kernel seriale a una versione vettorizzata e cache-friendly; saper parallelizzare su nodo singolo con OpenMP gestendo correttamente la memoria condivisa; saper individuare bug di correttezza e colli di bottiglia seriali con strumenti standard.

Competenze specifiche in uscita

Il partecipante sa analizzare e ottimizzare codice seriale guidato da misure, e sa scrivere codice OpenMP corretto ed efficiente su architetture NUMA.