Abstract

This proposal will focus on code transformation for legacy sequential programs toward thread-level parallelism on many-core computers. It highlights a semi-automatic method: Scalable Task-Oriented Code Transformation (STOCT). STOCT introduces thread parallelism for sequential code via a series of code transformation steps. STOCT’s goal is: Given a large C/C++ software program, transform the source code to replace several independent copies of a sequential process with an equivalent single process consisting of several threads. This takes advantage of many-core computers in a memory-efficient and scalable manner.

The STOCT methodology is implemented in the parallelization work for three widely varying applications: (i) a simulation toolkit Geant4; (ii) a linear system solver based on ILU(k) preconditioning; and (iii) a large-scale data analysis tool AliRoot. STOCT bridges the gap from sequential programming to concurrent programming by decomposing the parallelization into four subgoals: to exert more parallelism, to share more data, to guarantee correctness, and to achieve scalability.

STOCT has the following features: (i) to pursue thread safety by maximal parallelism in which only the minimal shared data set is allowed; (ii) to reduce memory footprint by sharing some user-level data whose value is no longer changed after initialization; (iii) to guarantee runtime correctness, which is weaker and yet strong enough for production; and (iv) to achieve scalability by eliminating two serious performance bottlenecks particular for shared-memory computation: memory allocation and cache coherence miss.