Tessellation Operating System

Building a real-time, responsive, high-throughput client OS for many-core architectures

Juan A. Colmenares,[1] Sarah Bird,[1] Gage Eads,[1] Steven Hofmeyr,[2] Eduardo Huerta-Yero,[3]
Albert Kim,[1] Rohit Poddar,[1] Hilfi Alkaff,[1] Krste Asanović,[1, 3] and John Kubiatowicz[1]
[1] Parallel Computing Laboratory, UC Berkeley
[2] Lawrence Berkeley National Laboratory
[3] International Computer Science Institute

Abstract

Tessellation OS is a new many-core operating system aimed at providing performance predictability to a simultaneous mix of high-throughput parallel, interactive, and real-time applications in an efficient, scalable manner. To achieve this goal Tessellation combines Space-Time Partitioning, Two-Level Scheduling, and Adaptive Resource Allocation. Space-Time Partitioning provides performance isolation and strong partitioning of resources among interacting software components, called Cells. Two-Level Scheduling separates global decisions about the allocation of resources to Cells from application-specific scheduling of resources within Cells. Tessellation's Adaptive Resource Allocation Architecture uses performance measurements along with the performance requirements of the applications to size the hosting Cells in a way that strives to allow real-time and interactive applications to meet their deadlines without sacrificing the performance of other applications.

This poster presents Tessellation's Cell model, software design, and resource allocation architecture along with proposed hardware mechanisms to aid Tessellation. Although Tessellation runs on existing multi-core systems (e.g., x86), hardware enhancements have the potential to significantly improve its performance and the fidelity of the Cell abstraction. This poster describes several examples of hardware enhancements, including memory hierarchy bandwidth partitioning, way- and bank-partitioned caches for greater memory system control, and hardware-acceleration for inter-cell message-passing communication.