Java on 1000 Cores: Tales of Hardware/Software Co-design

Cliff Click

Azul Systems

Azul Systems designs and builds systems for running business logic applications written in Java. Unlike scientific computing, business logic code tends to be very large and complex (> 1MLOC is *common*), display very irregular data access patterns, and make heavy use of threads and locks. The common unit of parallelism is the transaction or thread-level task. Business logic programs tend to have high allocation rates which scale up with the amount of work accomplished, and they are sensitive to Garbage Collection max-pause-times. Typical JVM implementations for heaps greater than 4 Gigabytes have unacceptable pause times and this forces many applications to run clustered.

Our systems support heaps up to 600 Gigabytes and allocation rates up to 35 Gig/s with pause times in the dozen-millisecond range. We have large core counts (up to 864) for running parallel tasks; our memory is Uniform Memory Access (as opposed to the more common NUMA), cache-coherent, and has supercomputer-level bandwidth. The cores are our own design; simple 3-address RISCs with read- and write-barriers to support GC, hardware transactional memory, zero-cost high-rez profiling, and some more modest Java-specific tweaks.

This talk is about the business environment which drove the design of the hardware (e.g. why put in HTM support? why our own CPU design and not e.g. MIPS or X86?), some early company history with designing our own chips (1st silicon back from the fab had problems like the bits in the odd-numbered registers bleeding into the even-numbered registers), and finally some wisdom and observations from a tightly integrated hardware/software co-design effort.

S. Drossopoulou (Ed.): ECOOP 2009, LNCS 5653, p. 444, 2009.