A Maze Of Twisty Little Passages

In Crowther’s 1970s Colossal Cave Adventure, whose layout happened to be partly modeled after Kentucky’s Mammoth Cave, you may recall two mazes: the original “all alike” one and an “all different” one that was added later. The same kind of distinction is commonly made in classifying modern parallel computing systems as SIMD or MIMD, and providing different, often mutually incompatible, programming environments for each. Is it really necessary to make such a stark distinction between the two?

Consider the wooden maze (shown above) that was in our SC Research Exhibit at SC08-SC11. Each of the colored balls has a different path to take (MIMD), yet it is perfectly feasible to efficiently get all the balls to their respective destinations by a series of tilts of the table (SIMD).

GPUs (Graphics Processing Units). Modern GPUs are not exactly SIMD, using a model that avoids most scaling limitations of SIMD by virtualization, massive multithreading, and imposition of a variety of constraints on program behavior (e.g., recursion is not allowed by NVIDIA nor by ATI). This branch off the SIMD family tree has grown quickly, with new programming models and languages appearing at each new bud... but little code base and many portability issues. MIMD C, C++, or FORTRAN using MPI message passing or OPENMP shared memory are now the bulk of the parallel program code base, so we suggest using those – via the public domain MIMD On GPU (MOG) technologies we are developing.

SC09 MOG. Much more sophisticated analysis and transformations enabled mogasm to create a highly customized mogc for each program – making MOG execution nearly as fast as native CUDA. Slowdown was generally less than 6X and often just a few percent. Actually, there were over a dozen completely different approaches tried for mogasm to achieve this performance, including optimizations based on runtime statistics, scheduling using a genetic algorithm (GA), and even per-program automatic instruction-set recoding to improve runtime decode overhead.

SC09 MOG. The MOG environment using the best of the previous year’s interpretation strategies was released at http://aggregate.org/MOG/20101122 on November 22, 2010 as full “alpha test quality” source code. However, it was actually a complete re-write with one goal in mind: make it possible to trivially port various full-language compilers to target our system. This is accomplished using a new, accumulator plus registers, ISA for which a nasty set of scripts can retarget generic MIPS code. The GCC-based version is called mogcc, and can process any of the languages that compiler supports. The new ISA enables more optimizations than the old one, and hence typically outperforms it by a small margin. The assembler, mogas, generates an optimized CUDA interpreter named mog.cu.

SC10 MOG. In addition to a few bug fixes, an OPENCL target and MPI system calls are being tested for release. Work has slowed dramatically due to lack of external funding....