Abstract

Challenged by serious power and thermal constraints, and limited by available instruction-level parallelism, processor designs have evolved to multi-core architectures. These architectures, many augmented with native simultaneous multithreading, are driving software developers to use multithreaded programs to exploit thread-level parallelism. While multithreading is well-known to introduce concerns of data dependency and CPU load balance, less known is that the uncertainty of relative progress of thread execution can cause patterns of I/O requests, issued by different threads, to be effectively random and so significantly degrade hard-disk efficiency. This effect can severely offset the performance gains from parallel execution, especially for I/O-intensive programs. Retaining the benefits of multithreading while not losing I/O efficiency is an urgent and challenging problem. We propose a user-level scheme, iHarmonizer, to streamline the servicing of I/O requests from multiple threads in the OpenMP programs. Specifically, we use the compiler to insert code into OpenMP programs so that data usage can be transmitted at run time to a supporting run-time library that prefetches data in a disk-friendly way and coordinates threads' execution according to the availability of their requested data. Transparently to the programmer, iHarmonizer makes a multithreaded program I/O efficient while maintaining the benefits of parallelism. Our experiments show that iHarmonizer can significantly speed up the execution of a representative set of I/O-intensive scientific benchmarks.