An Efficient Multithreaded Implementation of a Reproducible Parallel Random Number Generator Using OpenMP

Haohai Yu, Yue Qiu, Michael Mascagni

Recent advances in processor architectures have lead to microprocessors than have multiple processor cores that permit the simultaneous execution of several independent computational threads. Providing mathematical software for these multicore/multithreaded architectures is a very interesting and important problem that has recently become more urgent. In particular, we are interested in extending the Scalable Parallel Random Number Generators (SPRNG) Library to this new multicore/multithreaded environment. In this paper we show how to incorporate the SPRNG library in a seamless way using the OpenMP mutlithreading environment. OpenMP is one of the most widely supported and easy-to-use shared-memory parallel programming tools. Currently, SPRNG parallelizes its random number generators through parameterization. This is in contrast with many parallel random number generation schemes that use cycle splitting. In this work we investigate using both parameterization and cycle splitting to implement an OpenMP version of SPRNG. By marrying SPRNG and OpenMP (we will call it OpenMP-SPRNG), our method successfully meets the requirements of this new multicore/multithreaded environment and guarantees the reproducibility of SPRNG. Furthermore, OpenMP-SPRNG preserves the easy-of-use of OpenMP itself. Our results show that the expected and almost perfect parallelization efficiency is achieved through both parameterization and cycle splitting in OpenMP-SPRNG with very little difference in performance. While we have relied on parameterization in the past, the use of cycle splitting at the multicore/multithreaded level while still using parameterization for providing "top level" generators for the microprocessors seems to be both efficient and easy to conceptualize. We expect to provide these new capabilities in an upcoming version of SPRNG.