Reproducible Parallel Floating-Point Computations
Hong Diep Nguyen
James Demmel

Because of rounding errors, floating-point operations such as addition and multiplication are not associative, computed results depend also on the order of computation. Therefore we cannot get the same answer from run-to-run even on the same machine with varying number of available processors. That makes understanding the reliability of output harder, especially with the increasing level of parallelism. In the specific field of testing and debugging for example, on one hand each run should produce the same trace as the previous one. On the other hand, the running time should not be too slow with regard to actual run. Our goals are to design reproducible algorithms which are as efficient as the non-reproducible counter-parts.

Currently, our works focus on designing repeatable algorithms on shared-memory multicore architecture, which is a subset of reproducibility yet still difficult to achieve the same performance as the non-repeatable counter-parts. Our first approaches include using reproducible reduction tree and using higher precision.

A set of parallel algorithms use reduction tree to produce the final result. This reduction tree is subject to change with varying number of computing threads, hence is the source of non-reproducibility. Our first approach is based on reproducible reduction tree, which is fixed ahead of computing time and does not depend on available resources at run time. Input data are split into fixed-size chunks, and reduction tree is imposed over these chunks instead of threads. Experimental results showed that the overhead of computing time introduced by the reproducible reduction tree is negligible in comparison with the conventional reduction tree.

Non-reproducibility of floating-point computation comes from rounding-errors, hence another solution to that might be to eliminate these rounding errors. Using a higher precision reduces the effect of rounding errors and might produce reproducible results given that the problem is not too ill-conditioned. Nevertheless, using a higher precision (for ex. double-double precision) increases drastically the computing time. Our objective is to obtain reproducible results not accurate results, it is not needed to use higher precision everywhere but only at some selective points of computations. This technique has been embraced in testing and debugging. Experimental results showed that using higher precision helps to provide reproducible results for some set of not too ill-conditioned problem with a minor overhead of computing time.