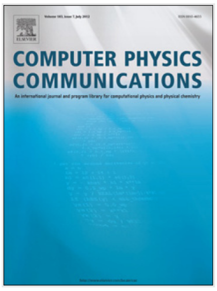

PII: S0010-4655(16)30128-X
DOI: http://dx.doi.org/10.1016/j.cpc.2016.05.012
Reference: COMPHY 5942

To appear in: *Computer Physics Communications*

Received date: 30 August 2014
Revised date: 3 May 2016
Accepted date: 6 May 2016

MPP–2014–327
PACS: 02.60.Jh

Concurrent Cuba

T. Hahn
Max-Planck-Institut für Physik
Föhringer Ring 6, D–80805 Munich, Germany

July 27, 2015

Abstract

The parallel version of the multidimensional numerical integration package Cuba is presented and achievable speed-ups discussed. The parallelization is based on the `fork`/`wait` POSIX functions, needs no extra software installed, imposes almost no constraints on the integrand function, and works largely automatically.

1 Introduction

Cuba is a library for multidimensional numerical integration written in C99 with interfaces for Fortran, C/C++, and Mathematica [1, 2, 3]. Cuba offers a choice of four independent routines for multidimensional numerical integration, Vegas [4], Suave [1], Divonne [5], and Cuhre [6], with very different characteristics.

Numerical integration is perfectly suited for parallel execution, which can significantly speed up the computation as it generally incurs only a very small overhead. Several features for concurrent sampling were added in Cuba versions 3 and 4, for both parallelization and vectorization. It was the objective to enable easy, ideally transparent, use of parallel Cuba, and led to the following design decisions:

1. No kind of Message Passing Interface is used, as that requires extra software to be installed. That is, the parallelization is restricted to one computer, using operating-system functions only. A standard setup these days is a single CPU with a number of cores, say 4 or 8. Utilizing many more compute nodes, as one could potentially do with MPI, is more of a theoretical option anyway since the speed-ups cannot be expected to grow linearly, see Sect. 4.2 on Performance.

2. Cuba uses `fork`/`wait` rather than the `pthread*` functions. The latter are slightly more efficient because parent and child share their memory space, but for the same reason they also require a reentrant integrand function, and the programmer may not have control over reentrancy in all languages (e.g. Fortran's I/O is typically non-reentrant). Also OpenMP, available with many compilers nowadays, has not been used since the Cuba library would depend on the way the integrand was compiled then. `fork` on the other