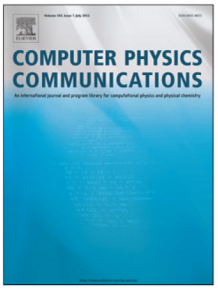
Accepted Manuscript



Massively parallel multicanonical simulations

Jonathan Gross, Johannes Zierenberg, Martin Weigel, Wolfhard Janke

PII: S0010-4655(17)30369-7
DOI: https://doi.org/10.1016/j.cpc.2017.10.018
Reference: COMPHY 6365

To appear in: *Computer Physics Communications*

Received date : 3 July 2017
Revised date : 12 October 2017
Accepted date : 20 October 2017

Please cite this article as: J. Gross, J. Zierenberg, M. Weigel, W. Janke, Massively parallel multicanonical simulations, *Computer Physics Communications* (2017), https://doi.org/10.1016/j.cpc.2017.10.018

Massively parallel multicanonical simulations

Jonathan Gross[a], Johannes Zierenberg[a,b,1], Martin Weigel[b,c], Wolfhard Janke[a,b]

[a] *Institut für Theoretische Physik, Universität Leipzig, Postfach 100920, D–04009 Leipzig, Germany*
[b] *Doctoral College for the Statistical Physics of Complex Systems, Leipzig-Lorraine-Lviv-Coventry ($\mathbb{L}^4$), D-04009 Leipzig, Germany*
[c] *Applied Mathematics Research Centre, Coventry University, Coventry CV1 5FB, England*

Abstract

Generalized-ensemble Monte Carlo simulations such as the multicanonical method and similar techniques are among the most efficient approaches for simulations of systems undergoing discontinuous phase transitions or with rugged free-energy landscapes. As Markov chain methods, they are inherently serial computationally. It was demonstrated recently, however, that a combination of independent simulations that communicate weight updates at variable intervals allows for the efficient utilization of parallel computational resources for multicanonical simulations. Implementing this approach for the many-thread architecture provided by current generations of graphics processing units (GPUs), we show how it can be efficiently employed with of the order of 10^4 parallel walkers and beyond, thus constituting a versatile tool for Monte Carlo simulations in the era of massively parallel computing. We provide the fully documented source code for the approach applied to the paradigmatic example of the two-dimensional Ising model as starting point and reference for practitioners in the field.

Keywords: GPU, parallel computing, Monte Carlo simulations, multicanonical, Ising model

PROGRAM SUMMARY

Program Title: cudamuca
Program Files doi: `http://dx.doi.org/10.17632/tzhfpdymv9.1`
Licensing provisions: Creative Commons Attribution license (CC BY 4.0)
Programming language: C, CUDA
External routines/libraries: NVIDIA CUDA Toolkit 6.5 or newer
Nature of problem: The program determines weights for a multicanonical simulation of the 2D Ising model to result in a flat energy histogram. A final production run with these weights provides an estimate of the density of states of the model.
Solution method: The code uses a parallel variant of the multicanonical method employing many parallel walkers that accumulate a common histogram. The resulting histogram is used to determine the weight function for the next iteration. Once the iteration has converged, simulations visit all possible energies with the same probability.
Additional comments including restrictions and unusual features: The system size and size of the population of replicas are limited depending on the memory of the GPU device used. Code repository at `https://github.com/CQT-Leipzig/cudamuca`.

Email addresses: `gross@itp.uni-leipzig.de` (Jonathan Gross), `zierenberg@itp.uni-leipzig.de` (Johannes Zierenberg), `Martin.Weigel@coventry.ac.uk` (Martin Weigel), `janke@itp.uni-leipzig.de` (Wolfhard Janke)
[1] Present address: Max Planck Institute for Dynamics and Self-Organization, Am Fassberg 17, D-37077 Göttingen, Germany

1. Introduction

The age of regular rapid increases in serial computational performance of commodity hardware came to an end about 15 years ago. Nonetheless, Moore's law, predicting an exponential growth in the number of transistors in typical integrated circuits, continues to hold [1]. But now the additional transistors are used to form further parallel computational units, or cores, instead of speeding up single threads. Consequently, the number of cores available to researchers working on cluster machines or supercomputers is growing rapidly, calling for the parallelization of established computational approaches and algorithms or, where this is not efficiently possible, the proposal of alternative solutions to the given computational challenges [2]. In many cases, for instance for Molecular Dynamics or Navier-Stokes simulations, such strategies will utilize domain decompositions, where each processor or thread is assigned to the environment of a subset of particles or cells. For Monte Carlo simulations the same approach works well for simple local-update schemes, where the acceptance probability of a local change is independent of any possible updates in distant areas. For non-local updates such as cluster algorithms domain decompositions become more involved [3]. Replica-exchange Monte Carlo and similar schemes, on the other hand, are inherently parallel [4]. A further important class of methods, namely generalized-ensemble simulations such as the multicanonical [5, 6] and Wang-Landau [7] approaches, cannot be easily treated in the same way, however. There, the acceptance probability for each move depends on the cur-