Abstract

In the course of less than a decade, Graphics Processing Units (GPUs) have evolved from narrowly scoped application specific accelerators to general-purpose parallel machines capable of accommodating an ever-growing set of algorithms. At the same time, programming GPUs appears to have become trapped around an attractor characterised by ad-hoc practices, non-portable implementations and inexact, uninformative performance reporting. The purpose of this paper is two-fold, on one hand pursuing an in-depth look at GPU hardware and its characteristics, and on the other demonstrating that portable, generic, mathematically grounded programming of these machines is possible and desirable. An agent-based meta-heuristic, the Max-Min Ant System (MMAS), provides the context. The major contributions brought about by this article are the following: (1) an optimal, portable, generic-algorithm based MMAS implementation is derived; (2) an in-depth analysis of AMD’s Graphics Core Next (GCN) GPU and the C++ AMP programming model is supplied; (3) a more robust approach to performance reporting is presented; (4) novel techniques for raising the abstraction level without sacrificing performance are employed. This represents the first implementation of an algorithm from the Ant Colony Optimisation (ACO) family using C++ AMP, whilst at the same time being one of the first uses of the latter programming environment.

References
Accelerated Combinatorial Optimization using Graphics Processing Units and C++ AMP

- O. Nitica, "A Parallel Ant Colony Optimization Algorithm for the Travelling
Accelerated Combinatorial Optimization using Graphics Processing Units and C++ AMP

Salesman Problem: Improving Performance Using CUDA,


Accelerated Combinatorial Optimization using Graphics Processing Units and C++ AMP


Index Terms

Computer Science Languages

Keywords

C++ AMP Generic Programming GPU Programming MAX-MIN Ant System Parallel Meta-Heuristics