Optional Course Projects

EE664: Introduction to Parallel Computing

Dr. Gaurav Trivedi
Department of EEE, IIT Guwahati

Problem 1

Phase 1 : Parallelization of SGA code using OpenMP.
Phase 2 : Parallelization of OpenMP version of SGA on GPU using CUDA.

SGA (Genetic Algorithm) is an effictive single-objective optimization algorithm proposed in 1994. The source code of the algorithm is written in C which can be downloaded from my homepage. Using this code , you have to develope a parallel version of SGA using OpenMP. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups for different **testcases**.

Notes: Before attempting this problem, please make sure you have a good understanding of one of the famous algorithms, i.e. Genetic Algorithm (GA).

Problem 2

Phase 1 : Parallelization of NSGA II code using OpenMP.
Phase 2 : Parallelization of OpenMP version of NSGA II on GPU using CUDA.

NSGA II (Non-Dominated Sorting Genetic Algorithm II) is a modified version of NSGA proposed in 2002. The source code of the algorithm is written in C which can be downloaded from my homepage. Using this code , you have to develope a parallel version of NSGA II using OpenMP. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups for different **testcases**.

Notes: Before attempting this problem, please make sure you have a good understanding of one of the famous algorithms, i.e. Genetic Algorithm (GA).

Problem 3

Phase 1 : Parallelization of MOEA code using OpenMP.
Phase 2 : Parallelization of OpenMP version of MOEA on GPU using CUDA.

MOEA (Multi-objective Evolutionary Algorithm) is an efficient multi-objective optimization algorithm. The source code of the algorithm is written in C & C++, which can be downloaded from my homepage. Using this code , you have to develope a parallel version of MOEA using OpenMP. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups for different **testcases**.

Notes: Before attempting this problem, please make sure you have a good understanding of one of the famous algorithms, i.e. Genetic Algorithm (GA).

Problem 4

Phase 1 : Parallelization of PSO (Particle Swarm Optimization) code using OpenMP.
Phase 2 : Parallelization of OpenMP version of PSO on GPU using CUDA.

PSO is an efficient optimization algorithm. The source code of the algorithm is written in C, which can be downloaded from my homepage. Using this code , you have to develope a parallel version of PSO using OpenMP. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups for different **testcases**. Before starting, please go through the source code of PSO for any bugs. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of one of the famous algorithms, i.e. Particle Swarm Optimization (PSO).

Problem 5

Phase 1 : Parallelization of SA (Simulated Annealing) code using OpenMP.
Phase 2 : Parallelization of OpenMP version of SA on GPU using CUDA.

SA is one of the most premitive efficient optimization algorithms. The source code of the algorithm is written in C, which can be downloaded from my homepage. Using this code , you have to develope a parallel version of SA using OpenMP. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups for different **testcases**. Before starting, please go through the source code of SA for any bugs. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of one of the famous algorithms, i.e. Simulated Annealing (SA).

Testcases (for Problem 1-5)

1. Minimize functions $f_1(x)$ and $f_2(x)$, such that

   (a) $f_1(x) = x^2$

   (b) $f_2(x) = (x-2)^2$

   (c) Variable bounds $[-10^3, 10^3]$

   Optimal Solution : $x \in [0, 2]$

2. Minimize functions $f_1(x)$, $f_2(x)$ and $g(x)$, such that

   (a) $f_1(x) = x_1$

   (b) $f_2(x) = g(x)(1 - (\frac{x_1}{g(x)})^2)$

   (c) $g(x) = 1 + 9(\Sigma_{i=2}^{n} x_i)/(n-1)$

   (d) Variable bounds $[0, 1]$ & $n = 30$

   Optimal Solution : $x_1 \in [0, 1],\ x_i = 0\ (for\ i = 2, \ldots, n)$

3. Minimize functions $f_1(x)$, $f_2(x)$ and $g(x)$, such that

   (a) $f_1(x) = x_1$

   (b) $f_2(x) = g(x)(1 - \sqrt{(\frac{x_1}{g(x)})})$

   (c) $g(x) = 1 + 9(\Sigma_{i=2}^{n} x_i)/(n-1)$

   (d) Variable bounds $[0, 1]$ & $n = 30$

   Optimal Solution : $x_1 \in [0, 1],\ x_i = 0\ (for\ i = 2, \ldots, n)$

4. Minimize functions $f_1(x)$, $f_2(x)$ and $g(x)$, such that

   (a) $f_1(x) = x_1$

   (b) $f_2(x) = g(x)(1 - \sqrt{(\frac{x_1}{g(x)})} - \frac{x_1}{g(x)} sin(10\pi x_1))$

   (c) $g(x) = 1 + 9(\Sigma_{i=2}^{n} x_i)/(n-1)$

   (d) Variable bounds $[0, 1]$ & $n = 30$

   Optimal Solution : $x_1 \in [0, 1],\ x_i = 0\ (for\ i = 2, \ldots, n)$

Problem 6

Phase 1 : Parallelization of 2D Poisson's equation Solver, Using MPI.
Phase 2 : Parallelization of 2D Posisson's equation Solver using CUDA.

Poisson's equation

$$-\epsilon \bigtriangledown^2 \phi = \rho$$

ϕ is unknown and ρ is known

The solution of Poisson's or any other partial differential equation (PDE) is obtained by discretizing the equation over whole domain (geometry) into algebraic form. Later using numerical techniques, the solution of the partial differential equation is obtained. This problem uses Finite Difference method (FDM) for discretization. The source code can be downloaded from my homepage. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of PDE's, Poisson's equation, FDM.

Problem 7

Phase 1 : Parallelization of Finite Element Solution of 2D Poisson's equation using OpenMP.
Phase 2 : Parallelization of Finite Element Solution of 2D Poisson's equation using CUDA.

This problem is same as that of above one, except it uses Finite Element Method (FEM) to discretize PDE. The source code can be downloaded from my homepage. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of PDE's, Poisson's equation, FEM.

Problem 8

Phase 1 : Create test examples of Poisson's equation in FreeFem++ using MPI.
Phase 2 : Parallelization of FreeFem++ using CUDA.

FreeFem++ is a partial differential equation solver. It has its own language. FreeFem scripts can solve multiphysics non-linear systems in 2D and 3D. FreeFem++ is written in C++ and the FreeFem++ language is a C++ idiom. The source code of the library can be downloaded from **http://www.freefem.org/ff++/**. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision

Problem 8 continued on next page...

of speedups. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of using a C++ open source libraries and how to make changes into it.

Problem 9

Phase 1 : Create test examples of Poisson's equation in Fenics using MPI.
Phase 2 : Parallelization of Fenics using CUDA.

The FEniCS Project is a collection of free software with an extensive list of features for automated, efficient solution of differential equations. The source code of the library can be downloaded from **http://fenicsproject.org/**. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of using a C++ open source libraries and how to make changes into it.

Problem 10

Phase 1 : Create test examples of Poisson's equation in PETSc using MPI.
Phase 2 : Create test examples of Poisson's equation in PETSc using MPI-CUDA.

PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It supports MPI, and GPUs through CUDA or OpenCL, as well as hybrid MPI-GPU parallelism. The source code of the library can be downloaded from **http://www.mcs.anl.gov/petsc/**. Please add appropriate comments at each places of your code for proper understanding of the reviewer. Also the course project requires submission of a report on the algorithm and your work showing the comparision of speedups. If you find any bugs, debug it and please make a proper documentation about the errors or bugs corrected by you in your course project report.

Notes: Before attempting this problem, please make sure you have a good understanding of using a C++ open source libraries and how to make changes into it.