PARALLEL SYSTEMS PROJECT

CSC 548 HW4, under Dr. Frank Mueller

Submitted by: Kaustubh Prabhu (ksprabhu), Narayanan Subramanian (nsubram), Ritesh Anand (ranand).

Problem Description:

Assessing the benefits of CUDA accelerator on AMG2006 Benchmark of ASC Sequoia

AMG2006 is a parallel algebraic multi-grid solver for linear systems arising from problems on unstructured grids [1].

This benchmark is an SPMD C code which uses MPI [1] (extensively) and openMP constructs. There was a performance study done some time back on OpenMP for this application, but a full exploitation of threading is still to be done [2]. Due to this SPMD paradigm and its usage of data parallelism by forming logical sub-problem data grids we see tremendous scope for making use of NVIDIA CUDA accelerator which bases its parallelism on similar lines.

NVIDIA CUDA is general purpose programming architecture that facilitates usage of parallel processing capabilities of a class of GPUs (mostly from NVIDIA) in general purpose programming rather than restricting it to graphics processing [3].

Through our extensive searching over World Wide Web, we have concluded that **none such CUDA implementation for the AMG2006 benchmark exists** that we are aware of at this point in time.

Towards Solution:

The benchmark has many applications within this like Laplace, 27pt, Stencil, Linear, Solver, etc. We observed that on doing a weak scaling, we could find some hotspots where there is scope for computational improvement. Some results with problem scaling on varied large number of processing element are available in the "amg2006.readme" [1]. We will be comparing those trends with what we observe employing CUDA.

Though our initial analysis using GNU profiling tool gprof [4] we observe that there can be many performance bottlenecks, as some compute intensive functions are consuming a lot of total application time.

This optimization will be done by porting those methods (excluding library and initialization ones) in CUDA.

For computing the 3D laplace on a cube and the 27-point stencil problem, we came up with a list of methods that consistently become the bottleneck of execution. On doing a code walkthrough, we found that they are composed of simple c code with iterative structures acting on huge data list. We believe such code blocks can be speeded up by using the concept of GPU threads, which is likely to result in a better performance.

Roadmap:

1.) Do a profiling for this application and gather the statistics. Observe trends and scope for optimization.
2.) Run the application for different problem sizes, different processor topologies, and different applications.
3.) List those functions which consume more computational time. Omit the library functions and the set up related functions.
4.) Study the functions and port them independently on CUDA.
5.) Integrate the ported functions with the application
6.) Run, profile and compare the results to observe trend and draw conclusions.

Additional information can be accessed from**: http://www4.ncsu.edu/~ranand/PS_HW4/index.html**

References:

1.) Read Me for AMG2006 available in AMG C source tar at: https://asc.llnl.gov/sequoia/benchmarks/#amg
2.) https://asc.llnl.gov/sequoia/benchmarks/AMG_summary_v1.0.pdf
3.) http://www.nvidia.com/object/cuda_what_is.html
4.) http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html