Parallel Implementation of Conjugate Gradient Method for Discrete Poisson Problems

Sahithi Rampalli
The Pennsylvania State University

Abstract

Discrete Poisson Equation is used in various scientific studies such as heat flow, computational fluid dynamics and more. The size of the grids on which the equation is applied can of the order of 10^6. Thus an efficient solver is necessary. We show an efficient parallel implementation of Conjugate Gradient (CG) on CPUs and GPUs.

Introduction

Finite difference numerical method is used to discretize the 2-dimensional Poisson equation. The size of the grids on which the equation is applied can go as large as 10s or 100s of 1000s. For such large matrix sizes, it is clearly essential to parallelize and optimize the solvers. Hence, an efficient implementation of CG which can be used in various physical applications is designed and analyzed. We show the performance of parallel CG implementation using OpenMP framework and GPU implementation using OpenACC framework.

Direct vs Iterative Solvers

The expensive matrix-vector multiplication in the direct solver (LDL^T factorization) is replaced by a simple stencil operation in CG.
The speedup of CG over the direct solver is shown in the table.

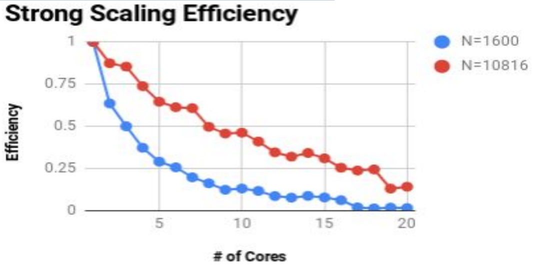
N	Speedup
64	4.8x
256	42.5x
1024	231x

Serial vs Parallel Codes

OpenMP framework is used to parallelize the CG implementation. Speedup of Parallel CG with 4 threads over serial CG is shown in the table.

N	Speedup
1024	2.1x
1600	2.2x
10816	3.3x

The trend in efficiency due to strong scaling for matrix dimensions 1600 and 10816 is shown in the plot below.

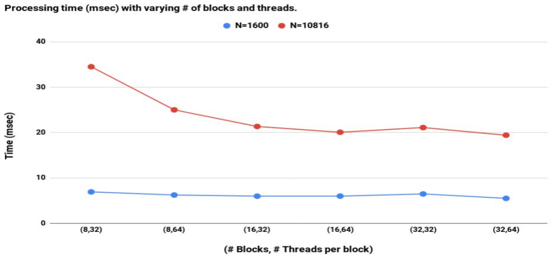


Acceleration on GPU

Using OpenACC frame work on CG, we executed CG on GPU. The speedup due to acceleration on GPU is expected to be higher. However, further optimization on data transfer between Host and Device has to be performed.
The trend in processing time by varying the number of blocks and threads per block can be seen in the adjacent graph.

Discussion & Conclusion

- Strong Scaling efficiency decreases with increase in number of cores.
- Speedup is greater for larger matrix sizes with the increase in number of cores.For production problem size of 250000 and 1000000, 10-14 cores would be optimal with an efficiency of about 0.5.



Contact:
Sahithi Rampalli
Penn State University
svr46@gmail.com:

Acknowledgement

I acknowledge Dr. Adam Lavely, Dr. Christopher Blanton for providing the required resources to make this work happen. I acknowledge ACI-ICS and PSC for providing the required servers.

References

1. https://arxiv.org/abs/1803.03797
2.