Parallelization of Poisson Equation Solver with Debye-Huckel's Linear Term

Yueze Tan
Department of Materials Science and Engineering, Pennsylvania State University

Problem of Interest

The problem is related with solving the electrostatic equation (Poisson equation) in a system with free moving charges and shielding effects, typical examples of which would be plasma or electrolytes. From Boltzmann's distribution we can obtain the equation describing the shielded, or compensated field:

$$(\nabla^2 - \lambda_D^{-2})\Phi = -\rho/\varepsilon$$

in which Φ is the electric potential, ρ is the charge density which is contributed by other effects than shielding, like charge density induced by inhomogeneous distribution of polarization in a dielectric matter, and ε is permittivity of the system. λ_D is known as Debye length, which is a term coming from Debye-Huckel's linearization of the electric shielding effect.

The problem could beconstructed as a linear equation ultimately, with the form of **A*x*** = ***b***, and

$$\mathbf{A} = \begin{bmatrix} \ddots & & \ddots & & \ddots & \ddots & \ddots & & \\ \cdots & \frac{1}{(\Delta z)^2} & \cdots & \frac{1}{(\Delta y)^2} & \cdots & \frac{1}{(\Delta x)^2} & C & \frac{1}{(\Delta x)^2} & \cdots \\ & & \ddots & & \ddots & & \ddots & \ddots & \ddots \end{bmatrix}$$

with

$$C \coloneqq -\left[\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} + \frac{2}{(\Delta z)^2} + \lambda_D^{-2}\right]$$

Serial: Direct and Iterative

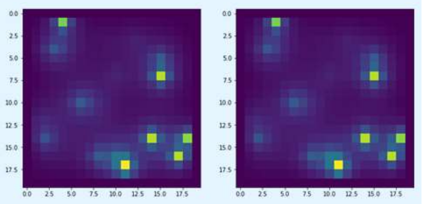
Both direct and iterative solvers are used for serial version of code, and only iterative part has been parallelized.

For direct solver, PLU decomposition is used, so that for a given linear system **A*x*** = ***b***, with **PA** = **LU**, we can solve x using

$$\boldsymbol{x} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}\boldsymbol{b}$$

The iterative solver makes use of Jacobi iterative method. Since the system matrix A is diagonally dominant, this iterative method converges unconditionally, and is reliable for most usual inputs. To get the result, the vector to be solved is updated iteratively:

A comparison between solutions for electric potential from direct (left) and iterative (right) solver is attached as below:



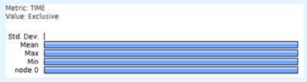
From which we can see the results are considerably close to each other.

To be filled with iterative error records

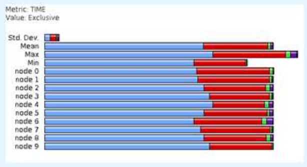
Parallelization: MPI

The parallelization method used in this project is MPI. Since the system matrix **A** could become considerably large for those product-size problems, it is a better practice to use a distributed memory method to carry out the parallelization.

For the serial code, the iterative solver itself costs around 100% percent of time:



Therefore the portion could be accelerated by parallelization is large for the solver. Profile of parallel version of code indicates the solving steps are still taking most of the time, and for large number of processes, the communication overhead is also obvious:



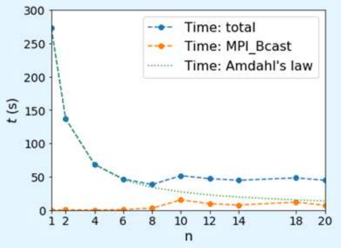
For the effect of acceleration, please refer to the "strong scaling" part. The speed up is considerably close to the theoretical limit, i.e., computational time decreases as $O(1/N)$.

Intentionally left blank

Intentionally left blank

Strong Scaling: Time v.s. Nprocs

For strong scaling study, we keep the problem size fixed with varying number of processes.



For number of processes not exceeding the number of cores on a single node, the scaling follows Amdahl's law in a considerably presice way:

$S(s) = t_{\text{serial}} / t_{\text{parallel}} = 1 / [(1 - p) + p / s] \approx s = N$,

with N as the mimimum number of processes. The speed of the code is limited ultimately by the computational resources available to the user. For test case shown, we have 8 cores on single node, and N = 8 reaches maximum of speed.

Acknowledgement

The bracket notation in calling of the matrix element and the stream output operator of matrix are inspired by the grammar from linear algebra template header library named Eigen: http://eigen.tuxfamily.org . The author appreciates using of these notations which make debugging a more concise process.

The lecturers have provided with a compiled version of tau with support of MPI, which makes profiling a more fluent work. The author would also like to thank their work on making this tool available.

Code Publication

The source code of the contents mentioned in this poster could be retrieved at Github: https://github.com/sunnyssk .

You could download the archive directly from the webpage.

References:

[1] Li, Y., Hu, S., Liu, Z., and Chen, L.-Q. Effect of electrical boundary conditions on ferroelectric domain structures in thin films. *Appl. Phys. Lett. 81*, 3 (2002).