Anqi Guo, Yunze Lian, Shineun Yoon

Richard Brower

Parallel Computing

May 1st, 2021

Solver Performance and Acceleration

Our problem and goal:

Our top priority is to achieve lots of solvers for linear equations, such as Jacobi, Gauss-seidel, multigrid and conjugate gradient and compare the performance of these solvers. We also use Openacc and open mpi to accelerate the algorithms.

As an acceleration of the iterative method, multigrid method (MG) and conjugate gradient (CG) has been discussed.

Equations to be solved:

We try to solve an heat flow problem with a 512*512 matrix A.

The equations to be solved are generally of the shape as below:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(t, x, y)$$

In this equation, u is the heat value to be solved for, t is the unscaled time, x and y are the unscaled spacial variables, and f is the source term determined arbitrarily.

Multigrid Method (MG)

Multigrid method (MG method) is an algorithm for solving differential equations using a hierarchy of discretizations. The problem we tried to solve in multigrid is a 2-dimensional time-dependent heat conduction equation. A non-zero initial heat field that varies with time is implemented in the middle of a 2D lattice grid with boundaries fixed at zero. The heat source changes based on sin function and the program simulates 100 units of time. In each time unit, the program simulates the heat flow until the whole 2D lattice grid settles.

The idea of multigrid is to separate the exact solution and error, and use a layered structure to accelerate the convergence.

$$u[t,i,j]_{exact} = u[t,i,j] + e[t,i,j]$$

U[t,i,j] is the approximate solution and e[t,i,j] is the error. The residue is

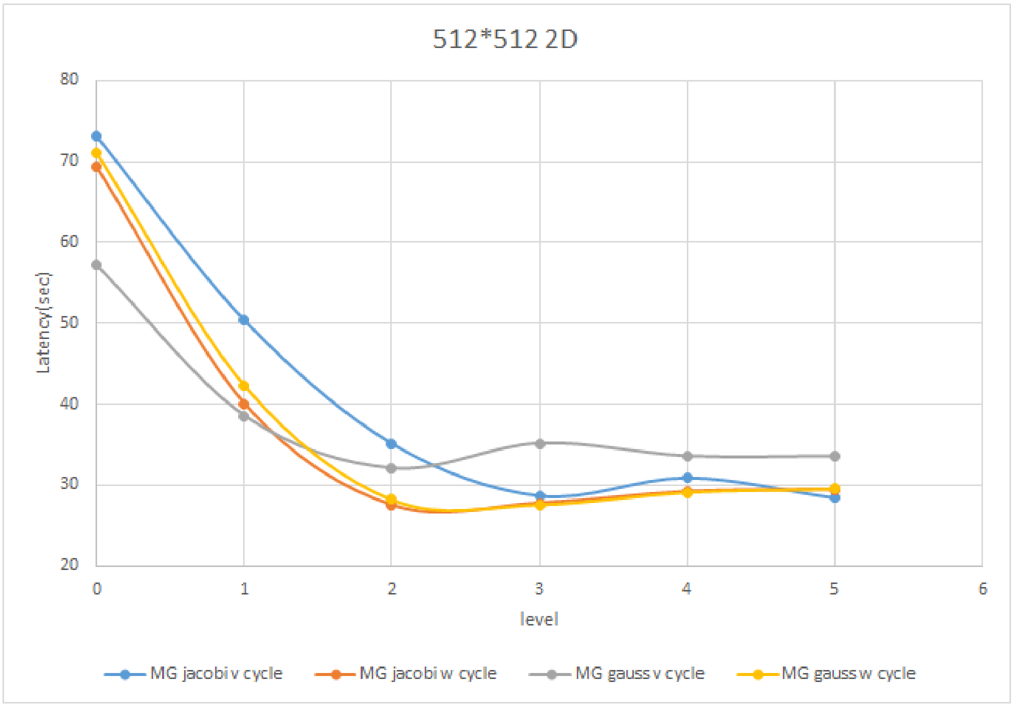
$$r[t,i,j] = f[t,i,j] - A^{t,i,j}_{t',i',j'} u[t',i',j']$$

We have:

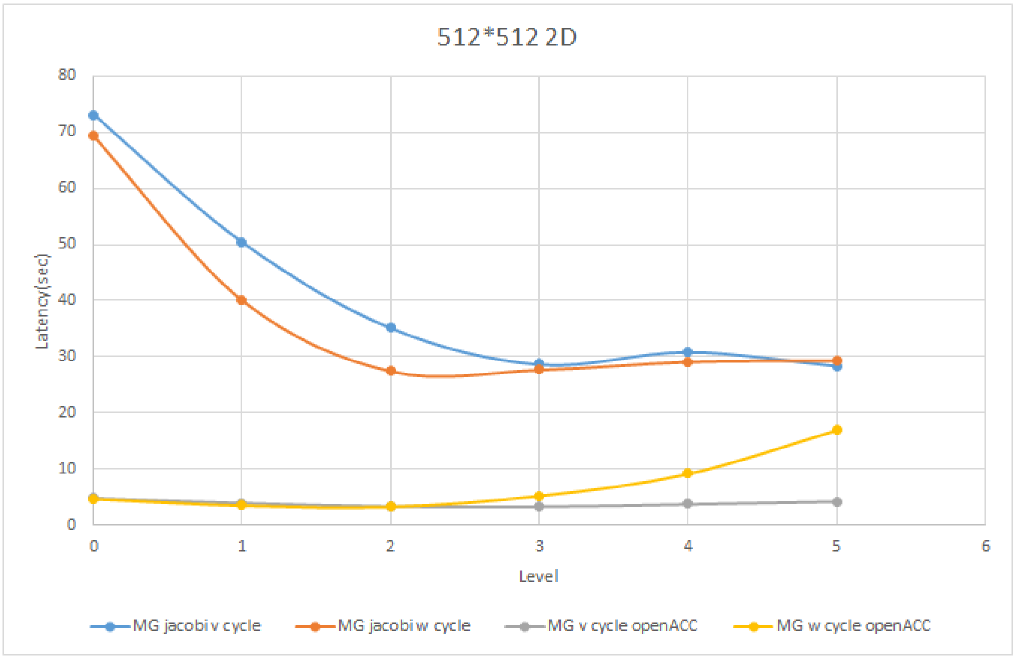
$$r[t,i,j] = A^{t,i,j}_{t',i',j'} u[t',i',j'] e[t',i',j']$$

Hence the error can be solved using the above equation. If the error is considered as the solution of the equation, there is error for the error as well. Therefore, the errors can be put in layers of different significance. As the layer gets deeper, the error becomes less significant. In that case, the deeper layers can be coarser without much loss in accuracy eventually. Once an error is approximately calculated, the corresponding solution gets corrected, and the correction is propagated to the layers above.

V-cycle and w-cycle multigrid method with two iterative solvers (Jacobi and Gauss-Seidel) performance is evaluated and accelerated with openACC using GPU.



The first figure shows the performance difference between v cycle and w cycle multigrid using jacobi and gauss seidel iterative solvers with different levels. The result shows the more level the better performance we will get. As level goes more than 3, each method will not get better latency. W-cycle with gauss seidel solvers gets the best performance when we have more than 2 levels.



The second figure is the performance comparison between MG serial code and openACC. We can see that openACC significantly accelerate the performance of MG. OpenACC gives us 6 times better performance when level is larger than 2. However, when level increases, the latency of MG w cycle with openACC increases which seems a little weird which might be bottlenecked by memory.

Conjugate Gradient (CG)

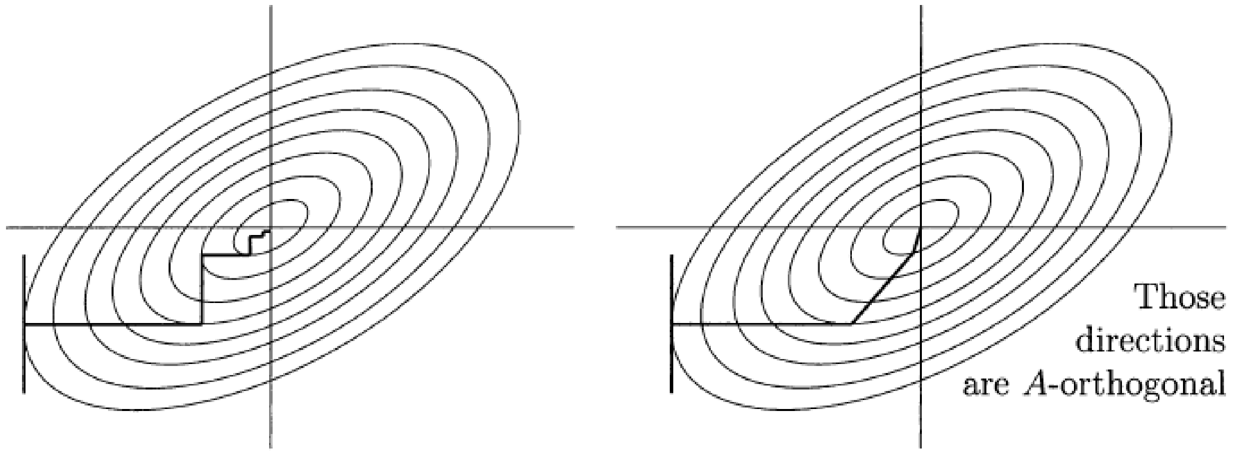


Figure 7.13: Steepest descent (many small steps) vs. conjugate gradients.

1. Idea of conjugate gradient

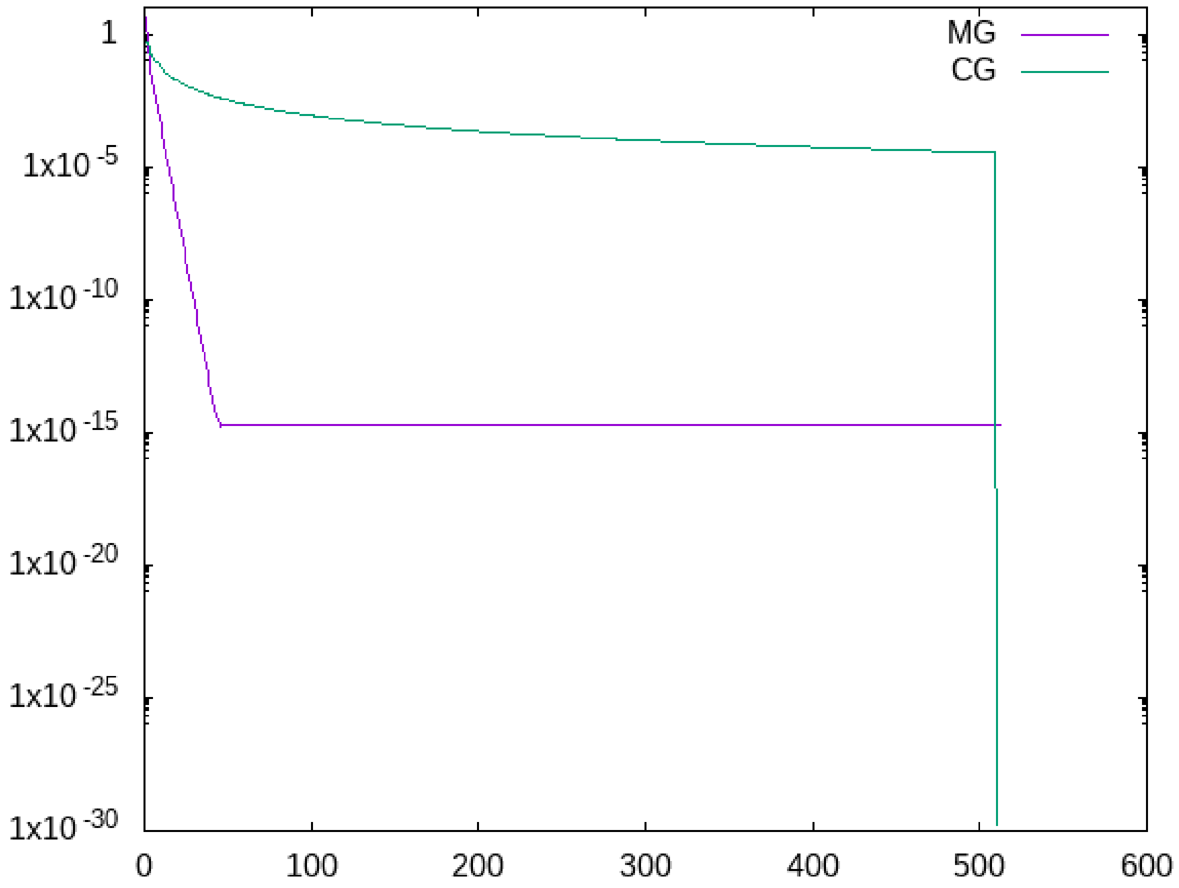
Solving the equation Ax=b is the same as getting the minimum point of the function recursively. Conjugate gradient method is just like an improved edition of gradient descent. When we use gradient descent, the X goes like the left figure. Go up and right, up and right for a lot of time. It could be faster if we just go right once and go up once, so we have a conjugate gradient method.

In CG method we try to let the direction d be orthogonal to error to make the error as small as possible. But in fact we can’t find such a d. (If we can find it, we could get the solution X directly). So instead we try to get a d which is conjugate orthogonal to the error. So we let the direction be conjugate orthogonal with all the errors. For a N dimension problem, we need to work at most N times.

2. Iteration formula of Conjugate Gradient.

```
1  $\alpha_k = r_{k-1}^T r_{k-1} / d_{k-1}^T A d_{k-1}$   % Step length to next $x_k$
2  $x_k = x_{k-1} + \alpha_k d_{k-1}$                 % Approximate solution
3  $r_k = r_{k-1} - \alpha_k A d_{k-1}$               % New residual from (14)
4  $\beta_k = r_k^T r_k / r_{k-1}^T r_{k-1}$          % Improvement this step
5  $d_k = r_k + \beta_k d_{k-1}$                      % Next search direction
```

3. Performance analyzing



This figure shows the relationship between iteration times(x axis) and residues (y axis) of MG and CG methods. The MG gets to the right answer very fast with few iterations. Besides, MG can have much smaller residues than CG methods.

Jacobi and Gauss-Seidel Methods

1. Relation between Jacobi and Gauss-Seidel

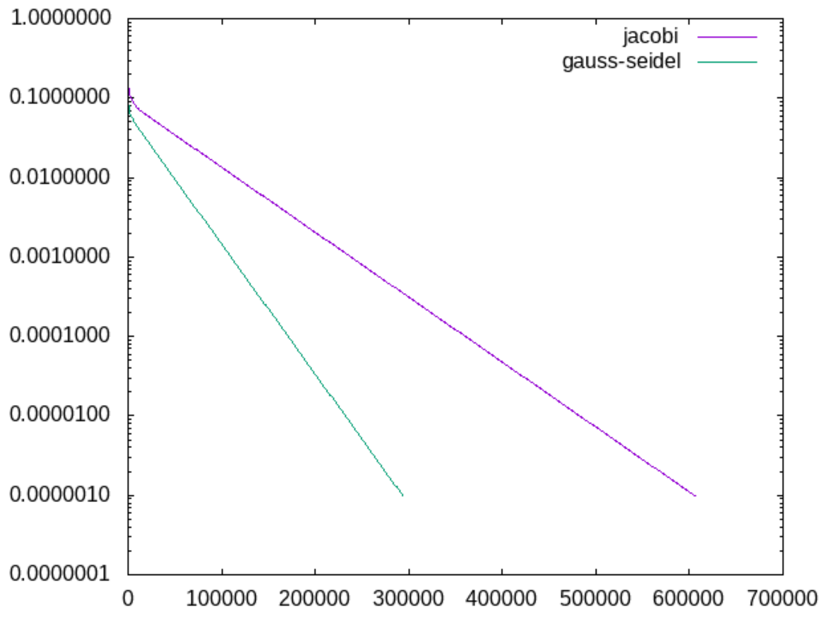
Rewrite $Ax = b \qquad Sx = Tx + b.$ 1

Pure iteration $\quad Sx_{k+1} = Tx_k + b.$ 2

$Se_{k+1} = Te_k$ which means $e_{k+1} = S^{-1}Te_k.$ 3

The basic logic of jacobi and gauss-seidel are the same. Using equation 1 to minus equation 2 then we can get equation 3.   Both Jacobi and Gauss-Seidel use a matrix ST to do linear transformation to error. The only difference between them is that they have different eigenvectors and eigenvalues for the matrix ST.

2. Performance analyzing



This figure shows the relationship between iteration times(x axis) and residues (y axis) of Jacobi and Gauss-Seidel methods. The speed of the jacobi and gauss-seidel depend on the eigenvalue of the matrix ST. The gauss-seidel is faster because the ST matrix of it has smaller eigenvalue.

References

1. On Solvers: Multigrid Methods. COMSOL Multiphysics. (n.d.). https://www.comsol.com/blogs/on-solvers-multigrid-methods/.

2. G.Strang,"Iteration methods and Preconditions", *Introduction to linear algebra, 4th edition,* p481,2009
3. G.Strang,"Krylov Subspaces and Conjugate Gradients", *Computational Science and Engineering,* p591-593,2007
4. J.R.Shewchuk,"The method of Conjugate Directions", *An introduction to the Conjugate Gradient Method without the Agonizing Pain,* 1994

Acknowledge of team member’s contribution

Yunze Lian: Achieve and write the codes of conjugate gradient methods, jacobi methods, gauss-seidel methods. Compare the performance of these methods, draw figures, and write the paper in these areas.

Anqi Guo: Achieve and write the code of Multigrid with heat problem using jacobi and gauss-seidel method in v cycle and w cycle. Also accelerate the code with openACC. Generate results comparing serial code and openACC version with different levels.