Parallel Finite Element assembly

Michael Wathen[*]

May 4, 2016

1 Introduction

Many industrial and geophysical scientific computing problems require discrete solving techniques for partial differential equations (PDEs). The two key components of a PDE solving are:

- Discretisation;
- Linear/non-linear solve.

For discretisation there are three main methods which are use finite differences, finite volumes and finite element methods (FEM). FEM discretises the model and represents the solution with respect to basis functions. Such methods rely heavily on variational methods from calculus of variations to approximate the solution.

The second key component is, the solving technique. Often, for large three-dimensional scientific computing problems it is not possible to do a direct solve for the system matrix. This is because such problems still have a large band width (lots of entries fair away from the diagonal), and hence the memory and time consumption is too large. Therefore, an iterative approach is required for these problems.

For a efficient FEM implementation the work to construct an $n \times n$ matrix is $\mathcal{O}(n)$. However, this is order for the solve is not necessarily true for all PDE models. For example, if the viscosity is small then it is known that for Navier-Stokes equations problem of solving the discrete system becomes harder. There is a huge amount of work that goes into the development for optimal ($\mathcal{O}(n)$) solvers for PDE; for example see [3, 4, 6].

[*]Department of Computer Science, The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada, mwathen@cs.ubc.ca.

For this project we will mainly focus on parallel finite element method within the `FEniCS` software framework [7]. We will show assembly times for both the Laplacian and magnetohydrodynamics equations. Since the Laplacian plays an integral roll in many PDEs, we will also look at a the timings for a parallel solve of it.

2 FEM in `FEniCS`

To understand the principle of parallel finite element methods, it is necessary to look at the sequential version. Given a certain cell, $\mathcal{T}_h$, with the local-to-global degrees of freedom mapping i_T then the generic finite element assembly is given as:

Algorithm 1 FEM assembly

```
1: A = 0
2: for $\mathcal{T} \in \mathcal{T}_h$ do
3:     (1) Compute $i_T$
4:     (2) Compute $A_T$
5:     (3) Add $A_T$ to $A$ according to $i_T$:
6:     for $i \in \mathcal{I}_T$ do
7:         $A_{i_{T(i)}} \overset{\pm}{=} A_{T,i}$
```

end

From this algorithm, we see that the general principle of the FEM is to loop through the cells of the discretised domain and calculate the local cell matrices. Then the local elements are mapped back into the global matrix, A. This therefore means that for certain global degrees of freedom require multiple reads and writes.

The algorithm for a basic parallel finite element method does not really vary from the sequential version but assembles over a partitioned mesh. Therefore, for an efficient parallel implementation an appropriate mesh partitioning is required. The partitioned mesh should distribute the minimises the inter-process communication costs between the degrees of freedom.

[8]

3 Results

In this section we will provide two examples. First, we consider a simple Laplacian where we look at both the assembly and solve times in parallel.

Second, we look at the parallel assembly time for the magnetohydrodynamics problem.

The two numerical experiment have been carried out using the finite element software `FEniCS` [7] together with `PETSc4PY` package (Python interface for `PETSc` [1, 2]) and the multigrid package `HYPRE` [5].

3.1 Laplacian

For our first example, consider Laplace's equation with non-homogeneous Dirichlet boundary conditions

$$\begin{aligned} \Delta u &= f \text{ in } \Omega, \\ u &= g \text{ on } \partial\Omega. \end{aligned}$$

MPI	DoFs				
processes	14,739	107,811	823,875	6,440,067	50,923,779
1	6.44e-01	2.69e+00	1.86e+01	1.49e+02	-
2	1.91e-01	1.38e+00	1.04e+01	7.92e+01	-
4	1.29e-01	1.05e+00	5.66e+00	4.19e+01	-
8	8.71e-02	5.31e-01	3.11e+00	2.32e+01	1.88e+02
16	1.16e-01	4.34e-01	2.12e+00	1.34e+01	9.94e+01
32	2.48e-01	3.17e-01	1.69e+00	1.20e+01	9.14e+01

Table 1: Laplacian assembly time for different degrees of freedom and MPI processes

MPI	DoFs				
processes	14,739	107,811	823,875	6,440,067	50,923,779
1	3.67e-01	4.50e+00	4.34e+01	3.93e+02	-
2	2.12e-01	2.90e+00	2.79e+01	1.92e+02	-
4	1.74e-01	1.46e+00	1.40e+01	1.21e+02	-
8	8.42e-02	1.18e+00	1.18e+01	9.24e+01	7.76e+02
16	8.28e-02	1.26e+00	9.11e+00	8.00e+01	6.71e+02
32	6.27e-02	9.35e-01	8.70e+00	7.66e+01	6.50e+02

Table 2: Laplacian solve time for different degrees of freedom and MPI processes

3.2 MHD

For our second example, we will consider an incompressible magnetohydrodynamics model with Dirichlet boundary conditions:

$$\begin{aligned}
-\nu\,\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p - \kappa\,(\nabla\times\boldsymbol{b})\times\boldsymbol{b} &= \boldsymbol{f} \qquad \text{in } \Omega, && \text{(1a)}\\
\nabla\cdot\boldsymbol{u} &= 0 \qquad \text{in } \Omega, && \text{(1b)}\\
\kappa\nu_m\,\nabla\times(\nabla\times\boldsymbol{b}) + \nabla r - \kappa\,\nabla\times(\boldsymbol{u}\times\boldsymbol{b}) &= \boldsymbol{g} \qquad \text{in } \Omega, && \text{(1c)}\\
\nabla\cdot\boldsymbol{b} &= 0 \qquad \text{in } \Omega. && \text{(1d)}
\end{aligned}$$

In a standard FEM fashion, we linearize around the current velocity and magnetic fields and introduce basis functions corresponding to the discrete spaces as in [10]. This yields the following matrix system:

$$\begin{pmatrix} F(u) & B^T & C(b)^T & 0 \\ B & 0 & 0 & 0 \\ -C(b) & 0 & M & D^T \\ 0 & 0 & D & 0 \end{pmatrix} \begin{pmatrix} \delta u \\ \delta p \\ \delta b \\ \delta r \end{pmatrix} = \begin{pmatrix} r_u \\ r_p \\ r_b \\ r_r \end{pmatrix}, \tag{2}$$

with

$$\begin{aligned}
r_u &= f - F(u)u - C(b)^T b - B^T p,\\
r_p &= -Bu,\\
r_b &= g - Mu + C(b)b - D^T r,\\
r_r &= -Db,
\end{aligned}$$

where $F(u) = A + O(u)$. The matrices are: F, the discrete convection-diffusion operator; B, a discrete divergence operator; M, the discrete curl-curl operator; D, a discrete divergence operator; and C, a discrete coupling term.

At this point there is no known parallel implementation of this discretisation for the MHD model with these specific elements used in [10]. In the recent paper [9], the authors look at an exact penalty form of the MHD model. This means that there discretisation yields a 3×3 linear system to be solved at each non-linear iteration.

4 Conclusion

References

[1] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes,

MPI	DoFs				
processes	20,381	148,661	1,134,437	8,861,381	70,045,061
1	4.82e+00	3.94e+01	3.04e+02	2.46e+03	-
2	3.08e+00	2.09e+01	1.58e+02	1.30e+03	-
4	1.50e+00	1.06e+01	8.49e+01	6.60e+02	-
8	7.88e-01	5.86e+00	4.65e+01	3.58e+02	2.81e+03
16	6.72e-01	3.33e+00	2.40e+01	1.88e+02	1.47e+03
32	6.66e-01	3.29e+00	2.35e+01	1.85e+02	2.01e+03

Table 3: MHD assembly time for different degrees of freedom and MPI processes

K. Rupp, B. F. Smith, and H. Zhang. PETSc User's Manual. Technical Report ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[2] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2014.

[3] J. Bosch, d. Kay, m. Stoll, and A. Wathen. Fast solvers for cahn–hilliard inpainting. *SIAM Journal on Imaging Sciences*, 7(1):67–97, 2014.

[4] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, second edition, 2014.

[5] R. D. Falgout and U. Yang. *hypre*: A library of high performance preconditioners. In *Computational Science — ICCS 2002*, volume 2331 of *Lecture Notes in Computer Science*, pages 632–641. Springer Berlin Heidelberg, 2002.

[6] C. Greif and D. Schötzau. Preconditioners for the discretized time-harmonic Maxwell equations in mixed form. *Numerical Linear Algebra with Applications*, 14(4):281–297, 2007.

[7] A. Logg, K. A. Mardal, and G. N. Wells, editors. *Automated solution of differential equations by the finite element method*, volume 84 of *Lecture Notes in Computational Science and Engineering*. Springer, Heidelberg, 2012.

[8] F. Pellegrini. Scotch. `http://www.labri.fr/perso/pelegrin/scotch`.

[9] E. G. Phillips, H. C. Elman, E. C. Cyr, J. N. Shadid, and R. P. Pawlowski. A block preconditioner for an exact penalty formulation for stationary MHD. *SIAM Journal on Scientific Computing*, 36(6):B930–B951, 2014.

[10] D. Schötzau. Mixed finite element methods for stationary incompressible magneto–hydrodynamics. *Numerische Mathematik*, 96(4):771–800, 2004.