The Multi-Threaded Analysis of Multiscale Solver for Viscoelastic Fluids Based on Open FOAM

Yi Liu1, XiaoWei Guo[1*], Chao Li[1], Cheng Kun Wu[1], Xiang Zhang[1]and Canqun Yang[1]
[1]*College of Computer, National University of Defense Technology, No. 109 Deya Street, Changsha, China*

Keywords: Multi-threaded, Multiscale, Open FOAM.

Abstract: In this paper, we gave performanceanalysis of a multi-threaded multiscale numerical solver based on Open FOAM. In the multiscale solver, we find that the matrix-vector multiplication is not the most compute-intensive operations. The discretization of stochastic equation about the Brownian configuration fields consumes nearly half the time for simulation. Our analysis result could provide valuable guidance for the parallel performance improvement of multiscale solver based on Open FOAM.

1 INTRODUCTION

Multiscale modelling and simulations(Horstemeyer, 2009)are usually employed to capture important features of complex fluids at multiple scales of time or space. In the multiscale simulation of complex fluids, the macroscopic flow behaviours are intrinsically governed by the dynamics of microscopic molecules.And the micro-macromultiscale methods(Keunings, 2004)couple the coarse-grained molecular scale kinetic theory into the macroscopiccontinuum mechanics.A macroscopic computational model is fast and simple. Nevertheless, it fails to reproduce many complex phenomena observed in experiments since the macroscopic model ignores microscopic details of molecular dynamics. As a microscopic approach, the atomistic modelling could provide the most detailed descriptionof the fluids dynamics, thus leading to enormous calculations(Guo et al., 2016).However, the micro-macro multiscale methods directly employ kinetic theory models in flow simulations, avoiding potentially inaccurate closure approximations involved in macroscopic computational model. In addition, these techniques consume less computing resources than microscopic approaches.Therefore, the micro-macro methods for multiscale simulations have attracted great attention in recent years.

The Brownian configuration fields (BCF)(Hulsen et al., 1997) is one of the multiscale methods to model viscoelastic fluids, which made a breakthrough via the use of correlated local ensembles. The BCF method uses a uniform number of configuration fields at fixed spatial positions to model the dynamics of molecular chains. Not only does this method ensure a homogeneous polymeric density in physical space, but also avoids tracking discrete particles. In a BCF simulation, the motion of viscoelastic fluids is still governed by the Naiver-Stocks equation on the macro-scale while the viscoelastic stress is calculated through solving a large number of stochastic equations on the micro-scale and taking the ensemble average as the macroscopic result. We implemented a multiscale numerical solver using the BCF methods to exploring the molecular distributions based on Open FOAM(Open Source Field Operation and Manipulation)(Liu et al., 2018).

Open FOAM is an open source toolbox for solving particle differential equations through the Finite Volume Method (FVM)(Christopher and Greenshields, 2015).It is written in C++ and offers a flexible framework for users to customize and extend its existing functionality freely. The users could focus on the mathematical models without considering the underlying implementation in detail because of the sufficient abstraction provided by Open FOAM. Therefore, Open FOAM is widely accepted in both academic and industrial CFD (Computational Fluid Dynamics) communities.

Open FOAM is parallelized with general parallel protocol MPI (Message Passing Interface). To increase the parallel performance of the Open FOAM solvers on the high-performance systems(Yang et al., 2017;Wang et al., 2017), a

hybrid multi-threaded solution was introducedby Liu(2011), which usesMPI between nodes and OpenMP directives inside the node.This method is intended to reduce MPI communications.

However, nearly all the previous studies of the parallel performance mainly focused on the existing solvers and seldom mentioned applications extended from Open FOAM. In this work, we apply this hybrid multi-threaded strategy on our multiscale BCF solver to improve the parallel performance on the high-performance systems. In the remaining of this paper, we first present the model and the environment in Section 2. Then, Section 3 introduced the optimization strategies and made numerical validation. At last, we concluded this work and draw our conclusions in Section 4.

2 MODEL AND EXPERIMENTAL CONFIGURATION

2.1 Mathematic Model

From the macroscopic perspective, the motion of the isothermal and incompressible fluid with density ρ could be described by the dimensionless momentum balance equation

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{Re}\beta \Delta \boldsymbol{u} + \frac{1}{Re}\nabla \cdot \boldsymbol{\tau}_p \quad (1)$$

and the continuity equation

$$\nabla \cdot \boldsymbol{u} = 0 \quad (2)$$

where $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{\tau}_p$ represent the velocity, the pressure, and the viscoelastic stress of the polymer fluid.

From the microscopic viewpoint, the viscoelastic stress is calculated by solving a large number of stochastic equations and the ensemble average is taken as the macroscopic result. The stochastic equations is given by

$$d\boldsymbol{Q} = \left(-\boldsymbol{u} \cdot \nabla \boldsymbol{Q} + (\nabla \boldsymbol{u})^{\boldsymbol{T}} \cdot \boldsymbol{Q} - \frac{1}{2De}\boldsymbol{F}(\boldsymbol{Q})\right) dt + \sqrt{\frac{1}{De}}\, d\boldsymbol{W} \quad (3)$$

and the viscoelastic stress $\boldsymbol{\tau}_p$ can be calculated by

$$\boldsymbol{\tau}_p \approx \frac{\alpha_{b,d}(1-\beta)}{De}\left(\frac{1}{N_{BCF}}\left(\sum_{i=1}^{N_{BCF}} \boldsymbol{Q} \otimes \boldsymbol{F}(\boldsymbol{Q})\right) - \boldsymbol{I}\right) \quad (4)$$

where $\boldsymbol{Q}$ is a stochastic process representing the dumbbell's configuration vector and $\boldsymbol{F}(\boldsymbol{Q})$ indicates the spring force.The 3-dimensional Gaussian Wiener process$\boldsymbol{W}$is used to model Brownian forces and $\boldsymbol{I}$is a unity matrix. In Equation (4),$\alpha_{b,d}$specifies a spring dependent constant andN_{BCF}indicates the number of Brownian configuration fields.

From Equation (1) to (4), the dimensionless parameters De (Deborah number), Re (Reynolds number) and β(viscosity ratio)are defined as

$$Re = \frac{\rho_c U_c L_c}{\eta_s + \eta_p},\ De = \frac{\lambda U_c}{L_c},\quad \beta = \frac{\eta_s}{\eta_s + \eta_p} \quad (5)$$

where ρ_c, U_c, $L_c$represent the characteristic density, velocity and length in macroscopic flow. And η_s,η_p indicate solvent and polymeric viscosity, respectively.

2.2 Experimental Configuration

The simulation domain of thedesigned solveris a two-dimensional rectangular box. The viscoelastic fluid flows into the left boundaries and out of the right boundaries. The flow geometry is shown as Figure 1.

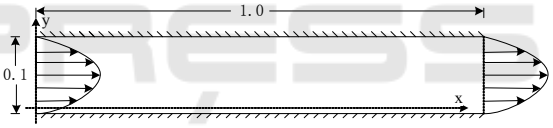

Figure 1: Flow geometry of viscoelastic fluids flow.

In this paper, we solve the BCF model with modified PISO algorithm in OpenFOAM. Gauss MINMOD and Gauss linear scheme are applied to discretize the spatial terms of the equation while Euler scheme is used for temporal discretization terms. After the partial differential equation is discretized using FVM, it can be transformed into a linear equation. Then we can use the iterative solvers provided byOpenFOAM to get the solutions at every time step. Preconditioned conjugate gradient (PCG) method is used to solve the pressure equation, and preconditioned biconjugate gradient (PBICG) method is for other equations(Liu et al., 2018).

3 OPTIMIZATION STRATEGIESAND VALIDATION

3.1 Optimization Strategies

The Open FOAM-based viscoelastic solver is parallelized with MPI parallel communication interface.The flow chart of MPI programming model in Open FOAM is shown in Figure 2.

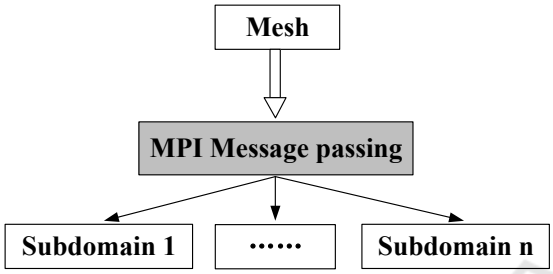

Figure 2: Flow chart of MPI programming in OpenFOAM.

In this model, one global mesh domain is decomposed into multiple load-balanced subdomains with the domain partition toolkit decompose Par in Open FOAM, which solved a big problem of MPI program. However, each MPI processjust handles the subdomain on their local processor. They have to rely on the point-to-point communication to update boundary field and the MPI_Allreduce()global communication to calculate residuals when solving the local linear systems in PISO algorithm.The communication traffic is proportional to the number and the area of subdomains for a specific simulation. Therefore, the pure MPI-based viscoelastic solver would be easily jammed by sending and receiving large number of messages(Culpo, 2011). In such case, a multi-level parallelism modelincorporating MPI and OpenMP as in Figure 3 is introduced to exploit more potential data parallelism in local MPI process(Zou et al., 2014).

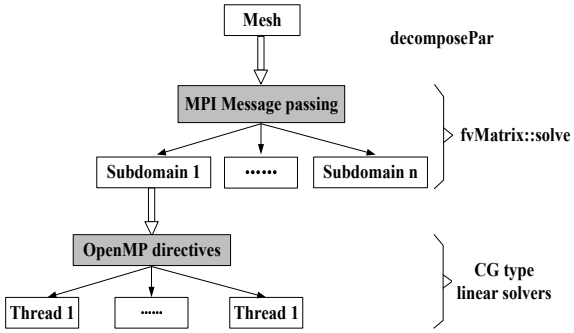

Figure 3: Flow chart of the hybrid parallelism model.

InFigure 3, each MPI process is assigned to an exclusive compute node to execute. Inside the compute node, the OpenMP threads are introduced to optimize the compute-intensive operations in conjugate gradient type linear solvers, such as cell-based loops and matrix-vector multiplication of Amul() and Tmul()(Paride Dagnaa, 2013). We added OpenMP directives "#pragma omp parallel for" into these operations without data dependency and distributed onto cores as independent threads inside the compute node. However, there is data race in original matrix-vector multiplication in OpenFOAM. To eliminate the inevitable data race, we reorganized the face-based loop to the cell-based loop as in Algorithm 1 for OpenMP multi-threading(Zou et al., 2014). The reorganized fvMatrix::Amul() function is given in Algorithm 1:

```
initMatrixInterfaces(…, psi ,…);
#pragma omp parallel for
for(label cell=0;cell<nCells;cell++)do
  scalar res=0;
  res=diagPtr[cell]∗psiPtr[cell];
  for(face owned by current cell)do
    res+=lowerPtr[face]∗psiPtr[uPtr[face]];
  end for
  for(face owned by neighbour cells)do
    res+=upperPtr[face]∗psiPtr[lPtr[face]];
  end for
  ApsiPtr[cell]=res;
end for
updateMatrixInterfaces(..., Apsi, ...);
```

3.2 Numerical Validation

We first test our numerical solver on the mesh (800×80) with the number of Brownian configuration fields $N_{BCF} = 800$ for 1 process with different number of threads.Each test has been run for 5000 time steps. All the tests are executed on a high-performance cluster with 390 nodes, each with

24 Intel Xeon E5-2692 CPU and 64GB of memory. The version of Open FOAM on the high-performance cluster is v4.0 and complied with GNU 5.4.0 compiler.

We reorganized the face-based loop to the cell-based loop and tested the wall time for 1 process with different number of threads of the multiscale solver. The results are shown in Table 1.

In order to comp are whether multi-thread method improves parallel performance more intuitively, we made a bar chart in Figure 4 with the data in Table 1 to show the speedup of different number of threads.

Table 1: The wall time for different number of threads of the multiscale solver.

Number of threads	Wall time(s)
1	2401
2	2367
3	2349
4	2354
5	2369
6	2363
7	2345
8	2356

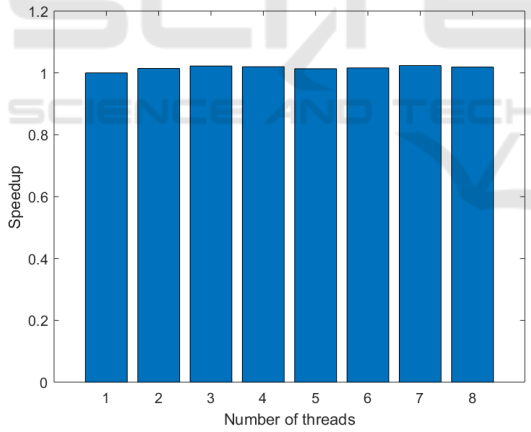

Figure 4: The total speedup of the multiscale solver with optimized matrix-vector multiplication for different number of threads.

As Figure 4 shows, the total speedup of the multiscale solver with optimized matrix-vector multiplication for different number of threads is almost equal to 1. It can be concluded that matrix-vector multiplication in conjugate gradient type linear solvers is not the compute-intensive operations in our multiscale BCF solver.

We test the average execution time for 5 main code segments of the multiscale solver with only 1 process. Furthermore, we test the average execution time of discretization and solving for $\boldsymbol{Q}$.The result of wall time is shown in Table 2.

Table2: Theaverage execution time for 5 main code segments in the multiscale solver run with only 1 process.

Code Segment	Execution Time(s)		
Initialization	673.863	Discretization	Solve
$\boldsymbol{Q}$	1212.09	958.13	253.96
$\boldsymbol{\tau_p}$	245.75		
$\boldsymbol{U}$	5.375		
PISO LOOP	186.312		

According to the data in Table 2, we draw a pie chart in Figure 5showing the time percent of each code segment in the multiscale solver. The time percent of discretization and solving for $\boldsymbol{Q}$ is also drawn in Figure 5.As Figure 5 shows, half of the average execution timeof the multiscale solver is used to solve the Brownian configuration field $\boldsymbol{Q}$. And the time percent of discretization for $\boldsymbol{Q}$ is 79% while solving for $\boldsymbol{Q}$ 21%. The matrix-vector multiplication would be only used when calling the function fv Matrix::solve() to solve these variables. Therefore, optimizing the matrix-vector multiplication contributes little to the parallel performance improvement.

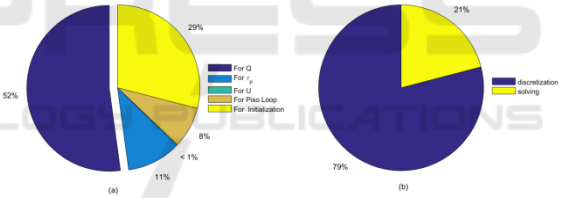

Figure 5: The time percent of (a) each code segmentin the multiscale solver; (b) discretization and solving for $\boldsymbol{Q}$.

4 CONCLUSIONS

Through the multi-threaded analysis of the multiscale numerical solver implemented with BCF method, we find that the compute-intensive part is not the matrix-vector multiplicationin conjugate gradient type linear solvers. The discretization of the Brownian configuration equations consumes nearly half the time for simulation. However, the large number of stochastic equations corresponding to Brownian configuration fields can be solved independently. In further study, we consider to distribute the stochastic equations onto multi-threads after mesh decomposition to improve the performance of the parallel algorithm.

ACKNOWLEDGEMENTS

This work was supported by the National Key Research and Development Program ofChina (No. 2016YFB0200401).

REFERENCES

1. HORSTEMEYER, M. F. 2009. *Multiscale Modeling: A Review*.
2. KEUNINGS, R. 2004. Micro-macro methods for the multiscale simulation of viscoelastic flow using molecular models of kinetic theory. *Rheology Reviews,* 2004**,** 67-98.
3. GUO, X.-W., XU, X.-H., WANG, Q., LI, H., REN, X.-G., XU, L. & YANG, X.-J. 2016. A Hybrid Decomposition Parallel Algorithm for Multi-scale Simulation of Viscoelastic Fluids. *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*.
4. HULSEN, M. A., HEEL, A. P. G. V. & BRULE, B. H. A. A. V. D. 1997. Simulation of viscoelastic flows using Brownian configuration fields. *Journal of Non-Newtonian Fluid Mechanics,* 70**,** 79-101.
5. LIU, Y., YANG, C., WU, C. K., ZHANG, X., ZHANG, X. & GUO, X. W. 2018. Exploring the Molecular Distributions in Dilute Polymer Solutions Using a Multi-Scale Numerical Solver. *Polymers.*
6. CHRISTOPHER, J. & GREENSHIELDS. 2015. *OpenFOAM Programmer's Guide* [Online]. Available: http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf [Accessed 05.09 2018].
7. YANG, X., WU, C., LU, K., FANG, L., ZHANG, Y., LI, S., GUO, G. & DU, Y. 2017. An Interface for Biomedical Big Data Processing on the Tianhe-2 Supercomputer. *Molecules,* 22**,** 2116.
8. WANG, W., YANG, X., YANG, C., GUO, X., ZHANG, X. & WU, C. 2017. Dependency-based long short term memory network for drug-drug interaction extraction. *Bmc Bioinformatics,* 18**,** 578.
9. LIU, Y. 2011. Hybrid Parallel Computation of OpenFOAM Solver on Multi-Core Cluster Systems. *Computer & Information Science*.
10. CULPO, M. 2011. *Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters* [Online]. Available: http://www.prace-ri.eu [Accessed 05.09 2018].
11. ZOU, S., LIN, Y. F., CHEN, J., WANG, Q. & CAO, Y. 2014. The Performance Analysis and Parallel Optimization of the OpenFOAM-Based Viscoelastic Solver for Heterogeneous HPC Platforms. *Applied Mechanics and Materials.*
12. PARIDE DAGNAA, J. H. 2013. *Evaluation of Multi-threaded OpenFOAM Hybridization for Massively Parallel Architectures* [Online]. Available: http://www.prace-ri.eu [Accessed 05.09 2018].