Hybrid Implementation and Optimization of OpenFOAM on the SW26010 Many-core Processor

Delong Meng[*], Minhua Wen[*], Jianwen Wei[*], James Lin[*†]

[*]Shanghai Jiao Tong University, Center for HPC

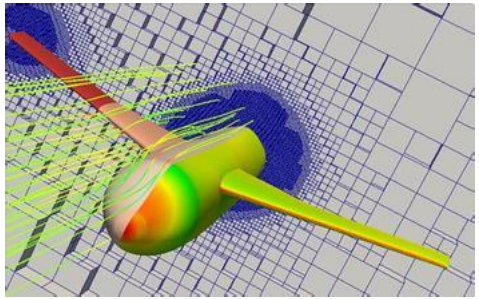
[†]Tokyo Institute of Technology, Japan

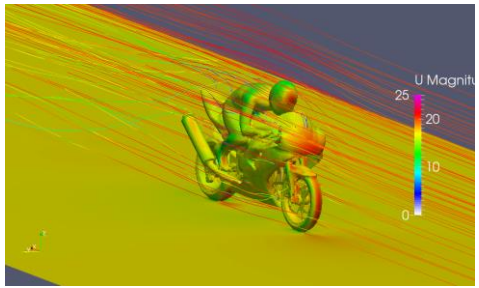
HPC China 2016, Xi'an China
October 27, 2016

Outline

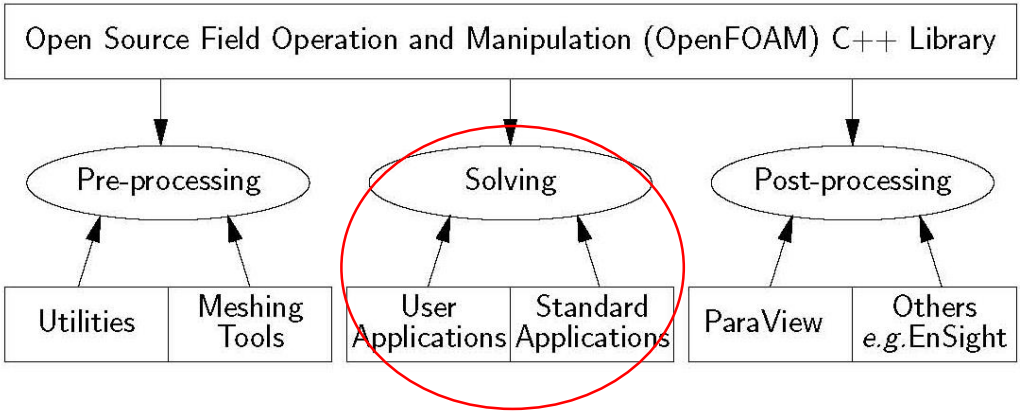
- Introduction
- Porting
- Methodology
  - Optimizations
- Result
  - Performance comparison step by step
  - Scalability evaluation on multi-nodes (Early Results)
- Conclusion

OpenFOAM





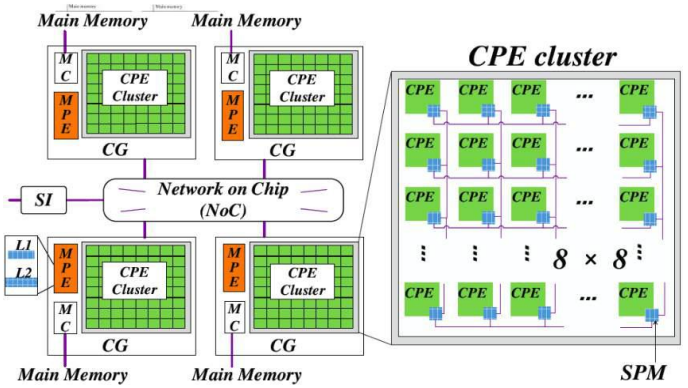
- OpenFOAM is a free software package written in C++, which has a large user base across most areas of engineering and science, for solving continuum mechanics problems. Solvers need to finish enormous computations on real datasets.
- It consists of a large set of pre-processing utilities, partial differential equation (PDE) solvers, and post-processing tools.

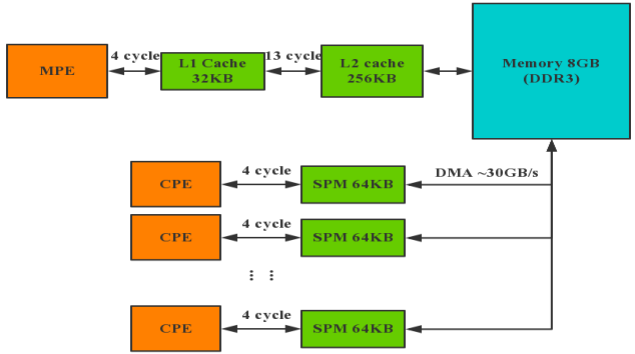


SW26010 Many-core Processor

- The Sunway TaihuLight Supercomputer
  - Peak performance: 125Pflops
  - Linpack performance: 93Pflops

- The SW26010 Many-core Processor
  - 4 core-groups (CGs)
    - 1 management processing element (MPE)
      - ✓ 23.2 Gflops
      - ✓ 32 KB L1 cache and 256 KB L2 cache
    - 64 computing processing elements (CPEs)
      - ✓ 11.6 Gflops * 64 = 742.4 Gflops
      - ✓ 64KB scratchpad memory (SPM) for each CPE





Goal, Proposal and Contribution

- **Goal**
  - Porting and optimizing OpenFOAM on Sunway TaihuLight.
- **Proposal**
  - Porting three basic solvers and ten incompressible solvers on the SW26010 Many-core Processor.
  - Optimizing the solvers on the MPE and achieving more than **2x** speedup .
  - Optimizing the solvers on the CPE cluster based on Sunway architecture.
- **Contribution**
  - The solvers achieve more than **2x** speedup on the MPE compared with the best performance tuned by compiler options.
  - The single-CG code runs more than **8x** faster than the optimized implementation on the MPE.
  - The porting method can be used for reference for other scientific programs.

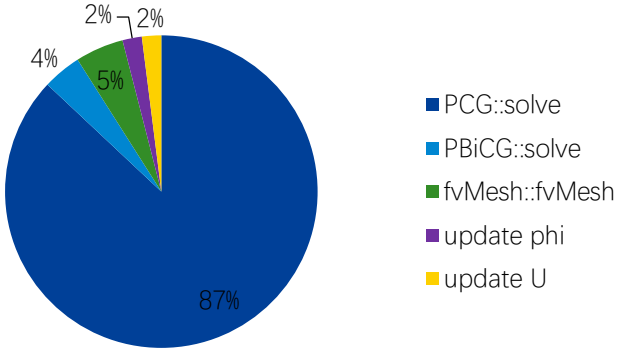
Outline

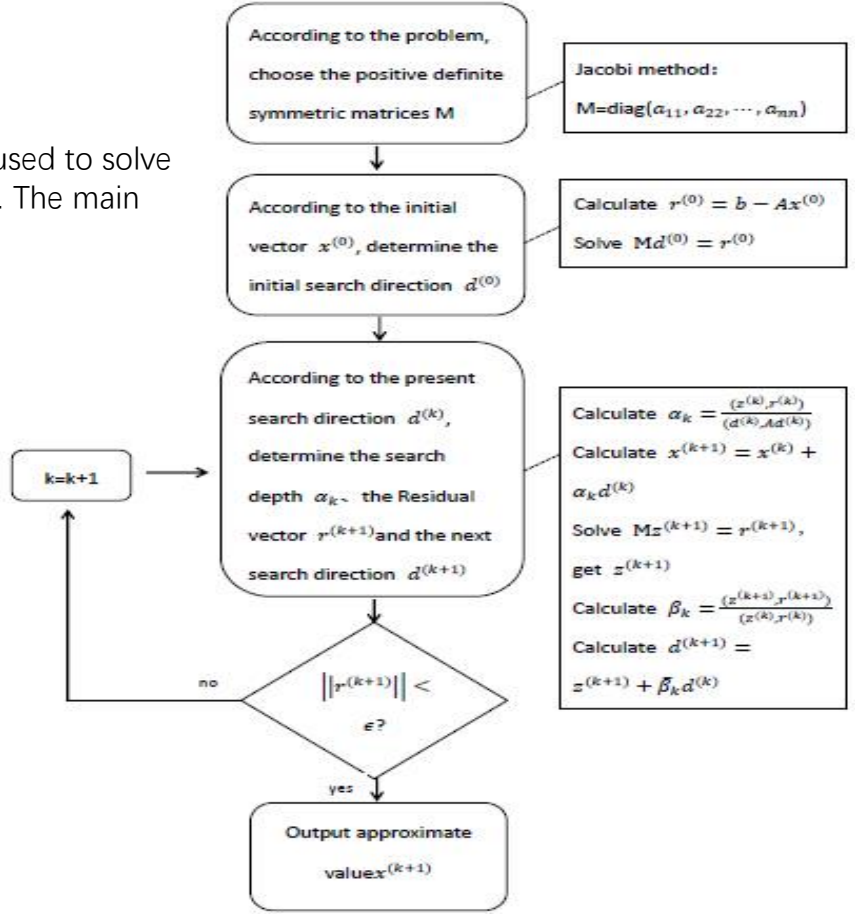
- Introduction
- Porting
- Methodology
  - Optimizations
- Result
  - Performance comparison step by step
  - Scalability evaluation on multi-nodes (Early Results)
- Conclusion

Key algorithm

Preconditioned Conjugate Gradient(PCG) is an important algorithm used to solve linear equations, the most expensive components of the selected solvers. The main building blocks of a PCG iteration:

- a sparse matrix vector multiply kernel
- an optional preconditioning
- three global dot products
- vector scaling
- vector-vector additions





Challenges

This is the first attempt to port OpenFOAM on TaihuLight supercomputer.

- **C++ programming language in OpenFOAM**
  - The CPE cluster only supports the C compiler sw5cc.
  - sw5cc is a high-performance compiler compared with the C++ compiler swg++.
- **Lack of dynamic library support on CPE**
  - OpenFOAM is divided into a set of precompiled libraries that are dynamically linked during compilation of the solvers and utilities.
- **Sparse matrix vector multiply kernel**
  - 64KB size limit of SPM for each CPE

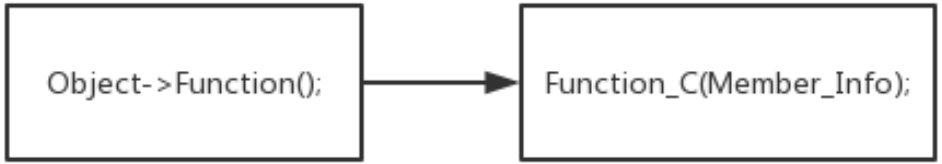
Rewrite the Kernel Code to C

The kernel code written in C is necessary for exploiting the computing power of the SW26010 processor.

Considering the overall complexity of the program, it is quite difficult to implement the program simply with C language.

We modify the data storage format and re-implement the kernel code with C language. The rest part of the program is still based on C++.

- a sparse matrix vector multiply kernel
- an optional preconditioning
- three global dot products
- vector scaling
- vector-vector additions

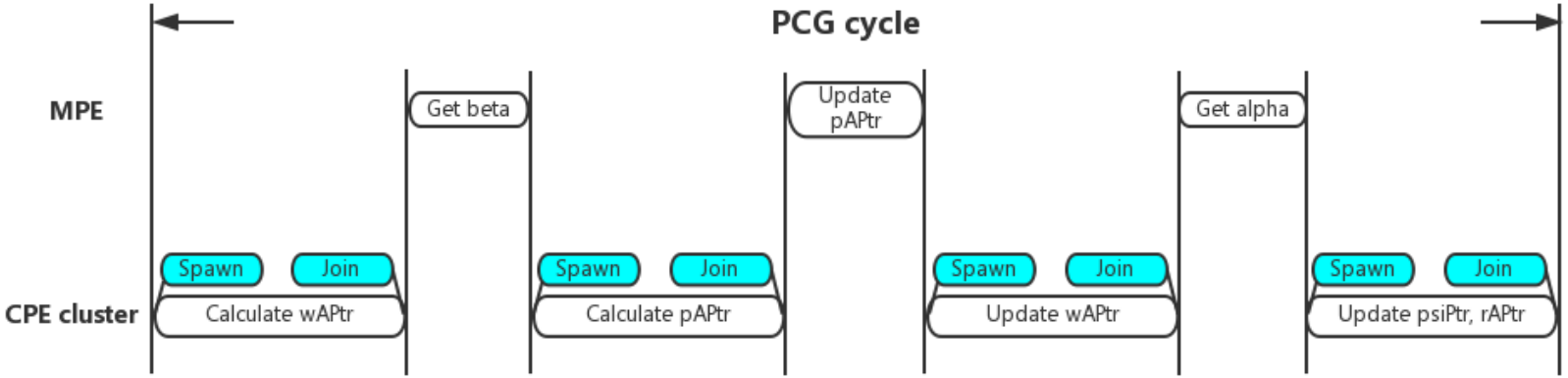


Porting to the MPE

- **Modify the profiles in wmake folder to compile OpenFOAM.**
  - libso->lib
- **--whole-archive and --no-whole-archive are applied on the libraries to generate the complete table.**
  - It is hard to find the linked libraries which are not clearly stated in the code. They are determined according to the input conditions.
- **Modify sw5cc compiler options to adjust the compiler behavior on the MPE.**
  - **-O3**
  - **-OPT:Ofast** is equivalent to -OPT:roundoff =3:Olimit=0: div_split=ON:alias = typed:.fast_math_library=ON.
  - **-LNO:prefetch_ahead=n** adjust the prefetching method of the compiler.
  - **-CG:pf_L1_ld:pf_L1_st:pf_L2_ld=0:pf_L2_st=0** can change the type of the prefetching instruction and determine the priority of the instructions.

Porting to the CPE cluster

- The athread library is designed for the MPE/CPEs programming model on the SW26010 processor.
- We develop the computation method on the CPE cluster with the athread library. Most of the computations are assigned to the CPE cluster and transfer data with DMA intrinsic.



Outline

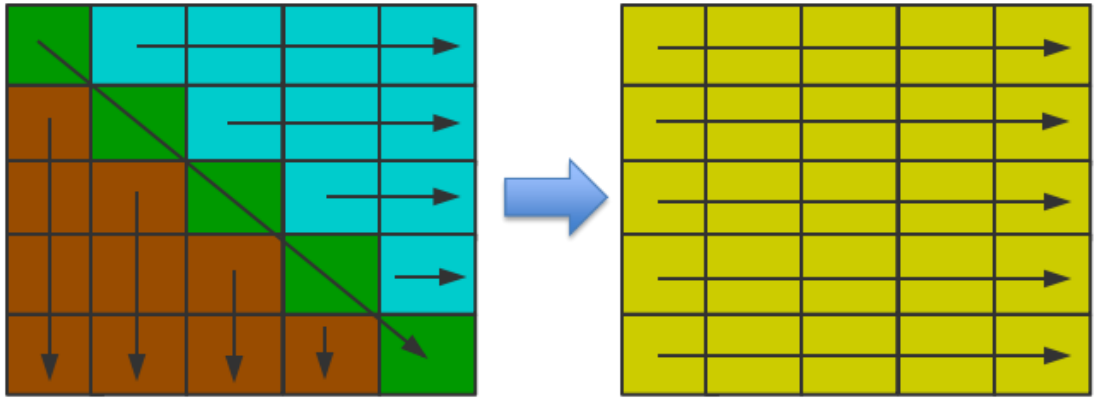
- Introduction
- Porting
- Methodology
  - Optimizations
- Result
  - Performance comparison step by step
  - Scalability evaluation on multi-nodes (Early Results)
- Conclusion

Optimizations on the MPE

- **Vectorization**
  - 256-bit SIMD intrinsic
  - Data blocking: finish all the related calculation before we update the data in the registers.
- **Data Presorting**
  - Presort data according to the column number of the sparse matrix
- **Elimination of the Redundant Computations**
  - Since the vectors are constant in the loop, the dot product can be accessed directly from the memory.
- **Loop Fusion**

Optimizations on the CPE cluster

- **Data Structure Transformation**

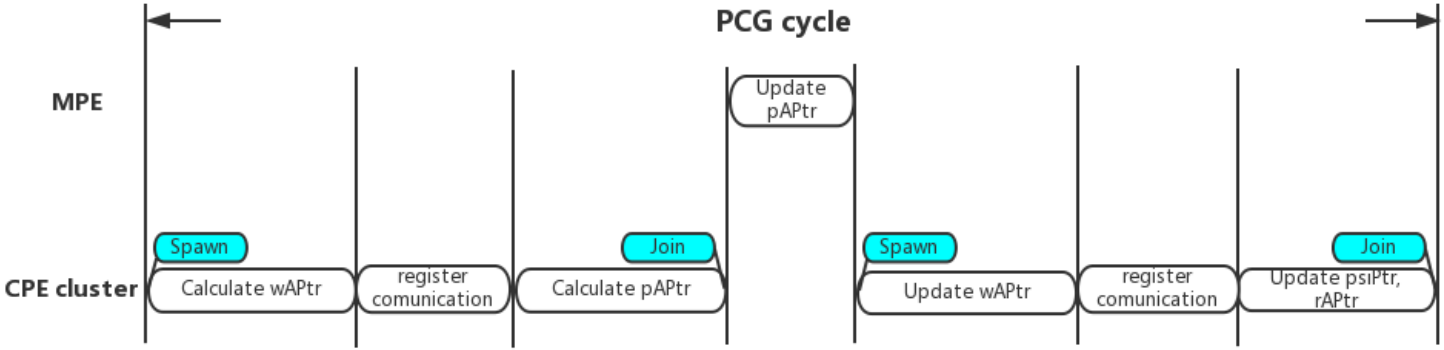


lduMatrix csrMatrix

- The coefficient matrix of OpenFOAM is stored in lduMatrix. Although the upperAddr is continuous, the lowerAddr is discrete. We use **csrMatrix** to store the data instead. With the new data structure, we calculate the search depth from the first row to the last row in order. Then each CPE updates the data continuously stored in memory and transmits one data block.

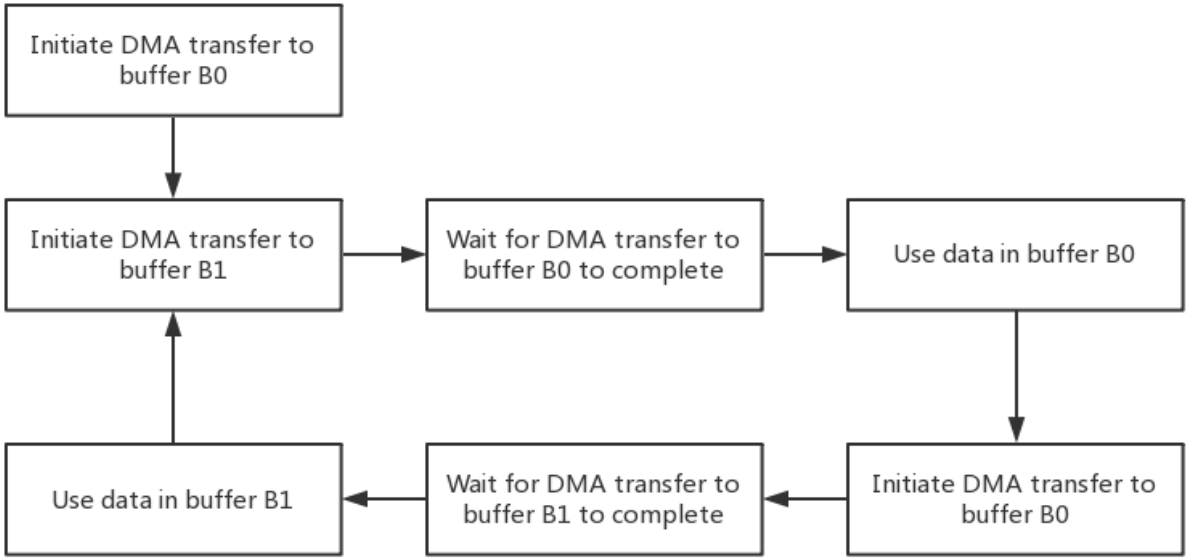
Optimizations on the CPE cluster

- **Register Communication**
- Register communication is designed for the communication between CPEs.
- Before we calculate the search direction, the present position, and the residual vector, we need to calculate the search depth with the whole matrix. It is distributed among the CPEs.
- In the register communication, each CPE starts with a buffer containing one element. The data from all CPEs are combined and accumulated at the first CPE within the same row/column into one buffer.



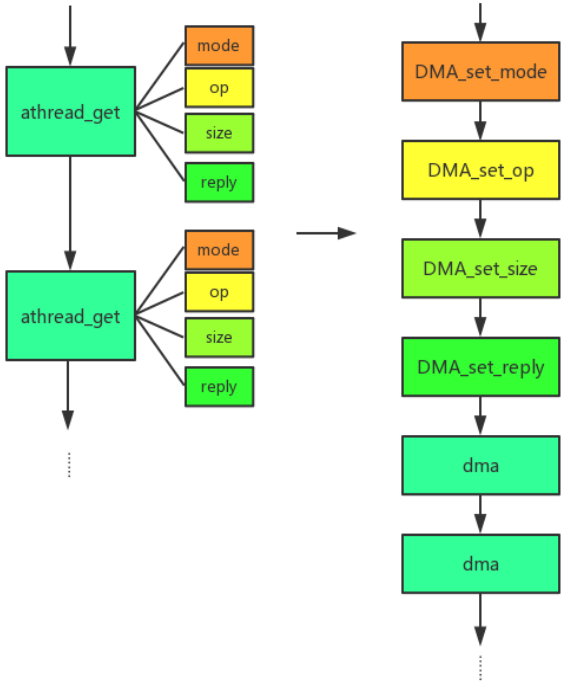
Optimizations on the CPE cluster

- **Double Buffering**
- Two buffers are used to hold a block of data. After double buffering, while the data is transmitted between memory and SPM, the data in the last loop can be computed in the meantime.



Other Optimizations on the CPE cluster

- **DMA Optimization**
  - Replace the memory access functions, athread_get and athread_put, with embedded assembly code and extract the invariants.
- **Prefetching**
  - By prefetching, register communication and memory access are overlapped during the loop. It improves the efficiency of the pipelines on the CPEs.
- **Data Reuse**
  - To further make use of SPM, we store the data on the boundary in the space of the input vector. Then we can store the data of the last block in the free space.

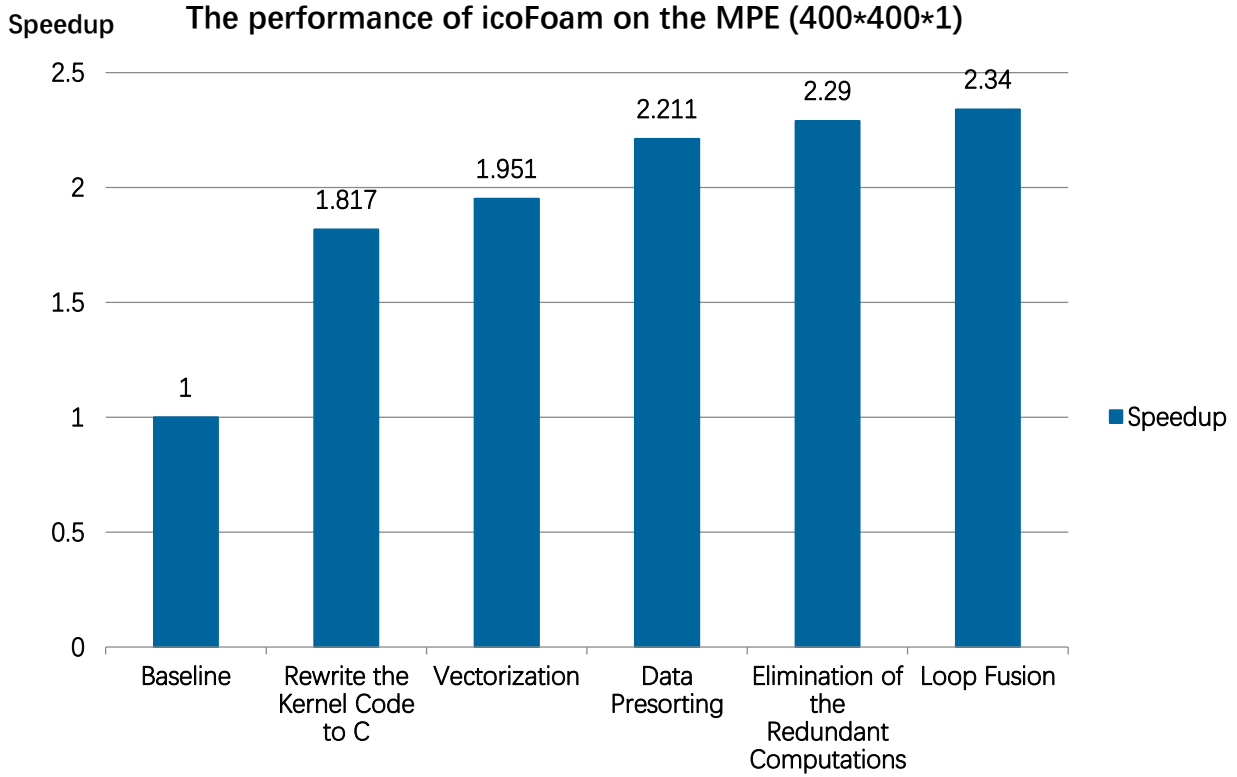


Outline

- Introduction
- Porting
- Methodology
  - Optimizations
- Result
  - Performance comparison step by step
  - Scalability evaluation on multi-nodes (Early Results)
- Conclusion

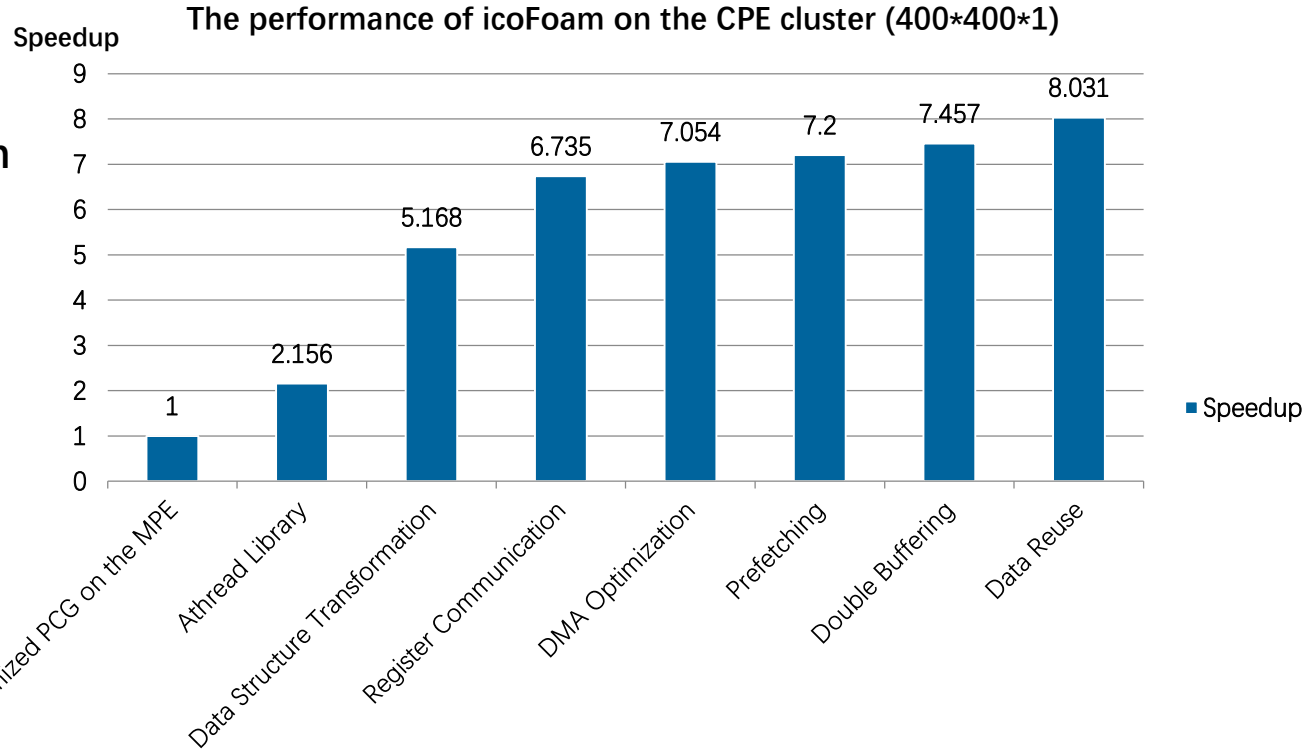
Test results on the MPE

We achieve 2.34x speedup on the MPE compared with the best performance tuned by compiler options.

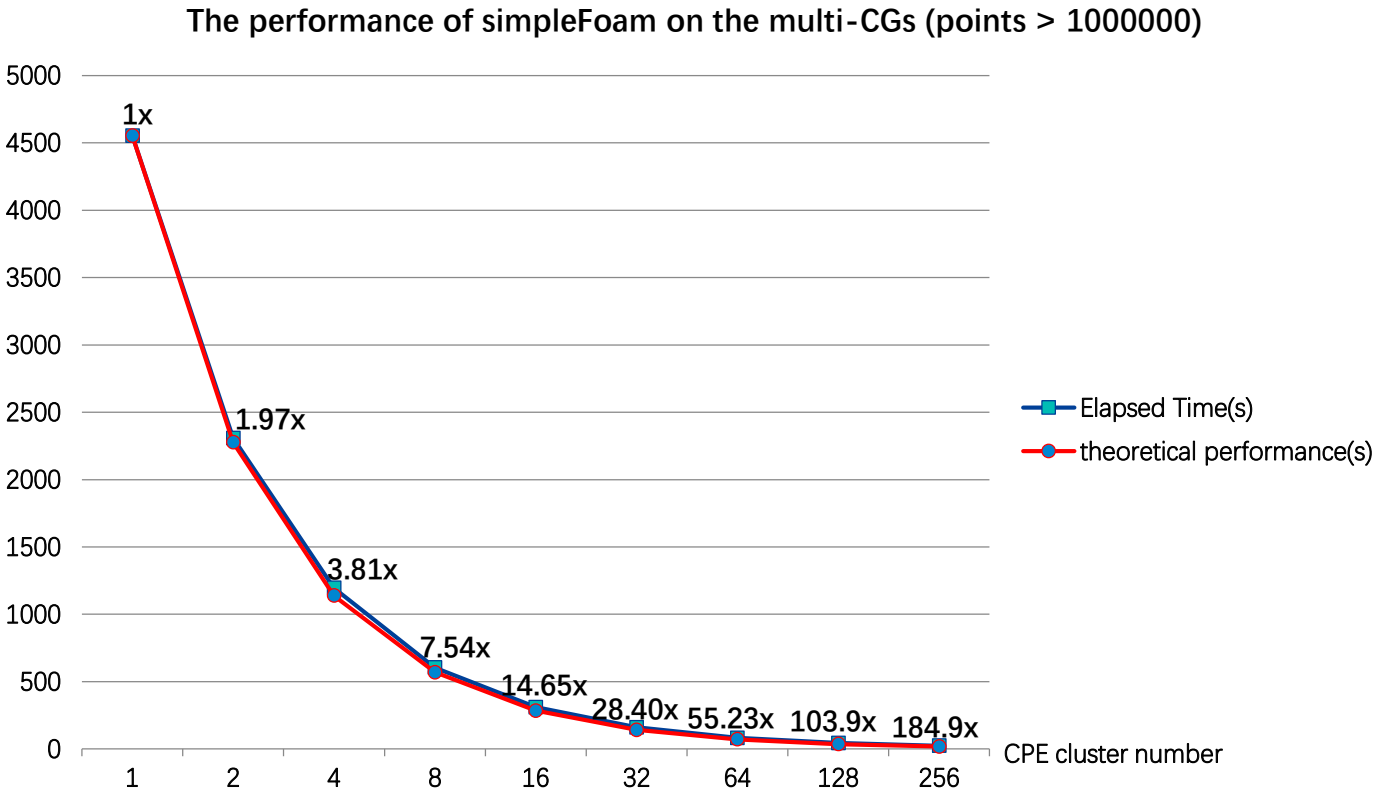


Test results on the CPE cluster

The single-CG code runs 8.03x faster than the optimized implementation on the MPE.



Strong scalability evaluation on multi-CGs (Early Results)



Outline

- Introduction
- Porting
- Methodology
  - Optimizations
- Result
  - Performance comparison step by step
  - Scalability evaluation on multi-nodes (Early Results)
- Conclusion

Conclusion

- The incompressible solver icoFoam achieves 2.34x speedup on the MPE compared with the best performance tuned by compiler options.
- The single-CG code runs 8.03x faster than the optimized implementation on the MPE.
- The linking mode of OpenFOAM can be used for reference to implement other programs based on the dynamic libraries.
- The mixed-language design with C and C++ can also be applied on complex C++ programs to utilize the computing power of the CPE cluster.

Future work

- Optimize MPI communication module to improve the efficiency of the solvers on the multi-nodes.

- Use register communication to share the present position among the CPEs and reduce the redundant memory access.

- Modify the assembly code of PCG and improve the efficiency of the overlap of computation and memory access with embedded assembly code.