ISSN 0554-6397
Izvorni znanstveni rad
(*Original scientific paper*)

Gordan Janeš
E-mail: gjanes@uniri.hr
Ante Sikirica
E-mail: ante.sikirica@uniri.hr
Center for Advanced Computing and Modelling, University of Rijeka,
Radmile Matejčić 2, 51000 Rijeka
Luka Grbčić
E-mail: lgrbcic@riteh.hr
Lado Kranjčević
E-mail: lado.kranjcevic@riteh.hr
Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka

MPI Associated Scalability of Open-Source CFD Codes for Oil Spill Assessment

Abstract

General-purpose CFD codes have recently become an increasingly discussed alternative to standardized, simplified and usually empirically calibrated specialized tools for pollution analyses. Commonly, CFD codes tend to provide physically more sensible results and can indicate the underlying cause for a given problem. Use for ecological problems, however, has usually been avoided due to the sizes of computational domains and inherent complexity of the calculations that need to be conducted. Adoption in recent years is mostly driven by significant improvements in computational capabilities and advancements related to code and communication optimizations. Unfortunately, due to substantial branching of codes and accompanying indispensable communication routines, especially in open-source community, performance and consequently applicability of codes, can vary significantly. This article aims to outline key limitations and quantify performance gains which can be obtained in a high-performance computing environment through the use of different communication protocols, when evaluating typical pollution problems such as oil spills. Obtained results indicate that savings of up to 40% in computational time can be achieved, depending on the code and message passing interface implementation for a problem in question, thus demonstrating the importance of communication protocols.

Keywords: MPI, CFD, oil spill

1. Introduction

Assessing oil spills and predicting their spread and extents on a larger scale is a complex problem, both numerically and in terms of physics/modelling. From an ecological standpoint, this topic is of great significance, especially in closed or semiclosed littoral regions, since the disturbance of the innate ecosystem can have long-lasting side-effects. This is the case of the Kvarner Bay, which is observed in this study.

In their study, Lončar et al. [1] modelled oil spills in the Kvarner Bay with MIKE 3 using Regional Ocean Modelling System (ROMS) data for the region in question. Wind influence on the overall circulation in the bay was also considered. Results are interesting, however, there are some obvious discrepancies between modelled and expected flow fields. Results presented in Ivić et al. [2] tackle a similar problem. Velocity field is obtained from MIKE 3 and subsequently advective-diffusive transport of a passive scalar (i.e. pollutant) was considered. It was shown that presented approach has merit. Spread of pollutants in the Bay was discussed in [3]. Similarly, MIKE 3 was used.

If we disregard the complexity of the interactions governing the spread of the oil spill, computational requirements are typically still great, especially for finite volume based CFD codes. However, due to generational improvements in code efficiency as well as increase in computational capabilities, CFD codes can be tuned in a manner which allows faster execution and hence enables their use for immensely large domains and complex problems. OpenFOAM has therefore been chosen for modelling of oil spills in Kvarner Bay, as it can be massively parallelised in high-performance computing (HPC) environment.

Scalability of OpenFOAM has been investigated by Culpo [4]. Results show acceptable speedups when using up to 1000 MPI processes for a given problem. The author has suggested further code improvements which could improve performance in HPC environments. In their study, Axtman and Rist [5] affirm that OpenFOAM can achieve ideal scaling when using up to 1000 processes. However, it is important to note that the underlying hardware plays a significant role. Observations presented in [6] suggest that OpenMPI offers performance improvements over Intel MPI.

In this study, data obtained from the Regional Ocean Modelling System [7] is used to establish a velocity field in the Kvarner Bay, which subsequently governs the spread of a randomly defined oil spill modelled as a passive scalar transport problem. Scalability of the presented problem is assessed on an HPC system with the goal of determining the optimal methodology for conducting similar simulations. Different variants of the OpenFOAM code as well as MPI libraries have been analyzed. By optimizing the use of computational resources for a given software package, data of interest can be generated in an acceptable time frame.

2. Methodology

2.1. Numerical model

Computational domain is a full-scale model of the Kvarner Bay, roughly 25044 × 22705 m in size, with a maximum depth of 77 m and an average of 60 m. The domain is modeled based on available geographical data and bathymetry measurements. Initially, three sets of grids, namely coarse, medium and fine, were generated through HEXPRESS. Numerical grids are fully hexahedral, generated by a successive increase in cell sizing with a factor of 1.4, and consist of $3.1 \cdot 10^6$, $3.8 \cdot 10^6$ and $7.1 \cdot 10^6$ cells respectively. Generated grids have an average y^+ value of 134.6.

Simulations are conducted in OpenFOAM. Variants maintained by the OpenFOAM Foundation Inc. [8] and OpenCFD Ltd. [9] have been considered. Identical (where applicable) case setups were defined for each distribution. An incompressible transient solver, pimpleFoam, has been used for flow modelling, whereas oil spill has been modelled as a simple scalar transport problem governed by the

$$\frac{\partial s}{\partial t} + \mathbf{u}\nabla s - D\nabla^2 s = 0 \tag{1}$$

where s is the scalar, $\mathbf{u}$ velocity field, t time and D diffusion coefficient. Diffusion coefficient can be calculated according to

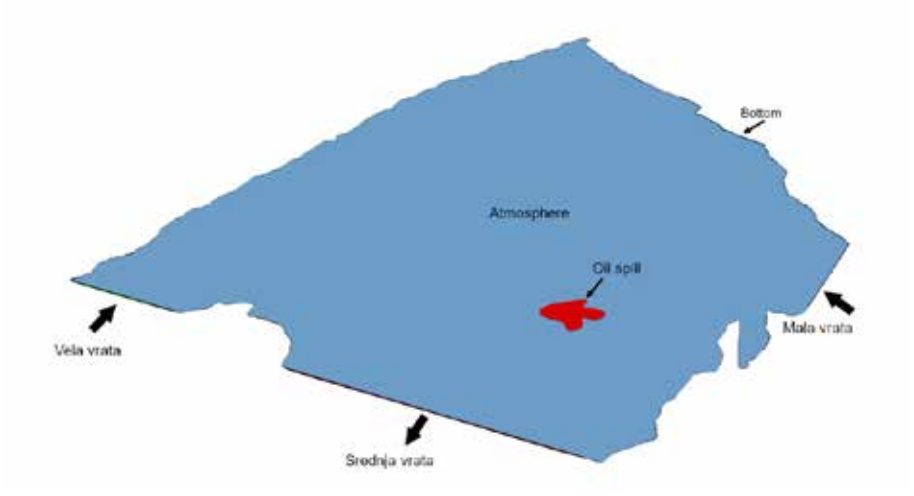
$$D = \alpha_1 \nu + \alpha_2 \nu_t \tag{1}$$

where α_1 and α_2 are constants and are equal to 10^{-9} and 0.714 respectively. ν_t represents turbulent viscosity. Kinematic viscosity ν is set to 10^{-6} m^2/s. Expression noted in (1) is implemented as an additional equation in the pimpleFoam solver to be calculated based on resolved (updated) velocity field. The k − ω SST model [10] is employed for turbulence modelling.

The problem in question is defined so that only the bottom of the bay is a fixed wall. Remaining patches, which correspond to atmosphere, passage Vela vrata and regions north of Srednja vrata and west of Mala vrata, are prescribed initial values or defined as outlets. ROMS [7] data is used as an input for boundaries in question. In total, 3721 data points can be extracted to model the region. These data points are interpolated and adapted to fit the observed domain.

For all tests, first order time discretization scheme is used. All divergence terms are prescribed second order schemes; linearUpwind, vanLeer and limitedLinear schemes are used for velocity, scalar and all remaining terms, respectively. Courant number is kept below 0.8 whilst residuals are set to 10^{-6} for pressure and velocity and 10^{-8} for the rest. Oil spill is defined as an arbitrary-shaped zone in the domain (on surface and extends 1 m below) with an area of roughly 2.23 km^2. An overview of the computational

domain, position and shape of the spill and boundary conditions, is given in Figure 1. Additionally, numerical grid is also shown.



(a)

(b)

Figure 1: Computational domain, spill and boundary conditions (a); details of the numerical grid when observed from top (left) and bottom (right) (b).

2.2. Testing environment

Evaluation matrix considers two OpenFOAM versions: OpenFOAM 8 (OpenFOAM Foundation) and OpenFOAM v2012 (OpenCFD). OpenFOAM 8 has been built with several MPI versions available. OpenMPI version 4.1.1 is used as the reference, with Intel MPI version 5.1.2 and version 2019 Update 8 additionally evaluated. Comparison with the OpenFOAM v2012 has been performed for the OpenMPI 4.1.1 implementation.

Evaluation is conducted on an Intel Xeon E5-2690v3 based cluster. Each node has 24 physical cores and 64 GB of RAM. Hyperthreading has been disabled in all scenarios and only 16 cores per node were used. Communication between nodes is achieved through the Infiniband FDR with a throughput of approximately 51 Gb/s. Use of the infiniband interconnect has been enforced throughout.

The overall impact of the MPI and general scalability have been assessed for 8 different core/node configurations; cases were evaluated when using 32, 64, 96, 128, 256, 512, 768 and 1024 cores.

3. Results and discussion

3.1. Grid convergence and numerical results

As noted, three grids have been considered for the simulations. Since one of the goals is to contrast the accuracy and performance i.e. achieve the optimal performance with acceptable accuracy, grid chosen for performance evaluation must exhibit balance between those aspects.

Accuracy has been determined based on consistency of results between several reference points in the domain. However, the most significance has been attributed to the overall congruency of the oil spill extents. These have been compared for concentration value $s = 0.01$ and, based on results, medium grid has been deemed appropriate for further analyses. This choice is appropriate since when using OpenFOAM 8 and Intel MPI 2019, whilst running on 48 cores, execution times for coarse, medium and fine grid are 52509 s, 35188 s, 110356 s, respectively. A comparison of spill's extents after 72 hours is shown in Figure 2.

Figure 2: Extents of the oil spill after 72 hours visualized for s = 0.01; coarse grid (black), medium grid (red) and fine grid (blue).

Spill extents after 7 days of simulation time are shown in Figure 3.b. Additionally, velocity field is given after 7 days (Figure 3.a). The evolution of the flow field governs the spill's movement. The overall approach with ROMS data nested in the OpenFOAM problem case produces convincing results.

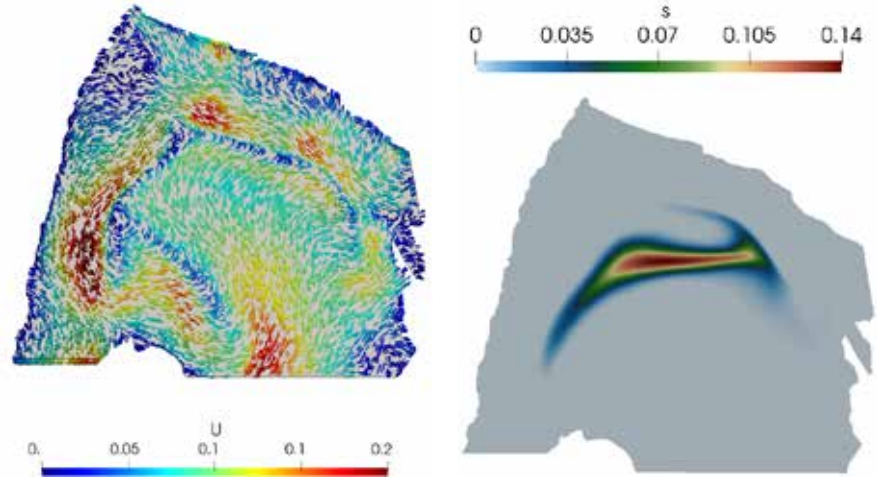


Figure 3: Results of the numerical simulations after 168 hours: velocity field after 168 hours (a); spill extents (scalar concentration) after 168 hours (b).

3.2. Performance and speedup

All performance assessment tests have been conducted three times. Averaged results are hence reported. Noted computational times correspond to simulations covering a 72-hour period. Results for a single-core-run are omitted from following tables (i.e. not considered), but will be included in the speedup calculations. Observations for different OpenFOAM distributions and MPI versions are given in Tables 1-4. Results include clock time, memory use and average time spent by the solver for the evaluation of each time step (TS).

Table 1: Scaling in OpenFOAM 8 when using Intel MPI version 2019 Update 8.

Cores	**32**	**64**	**96**	**128**	**256**	**512**	**768**	**1024**
Clock time [s]	60957	28222	18638	14232	8165	6452	6749	6841
Avg. time per TS [s]	1.097	0.512	0.338	0.269	0.155	0.114	0.117	0.123
RAM usage [GB]	7.839	11.391	15.468	19.431	37.801	45.105	68.370	90.839

Table 2: Scaling in OpenFOAM 8 when using Intel MPI version 5.1.2.

Cores	**32**	**64**	**96**	**128**	**256**	**512**	**768**	**1024**
Clock time [s]	61320	28183	18797	14478	8859	7478	8329	8501
Avg. time per TS [s]	1.105	0.511	0.341	0.270	0.164	0.130	0.147	0.146
RAM usage [GB]	5.584	6.909	8.141	9.656	15.780	24.957	36.865	52.760

Results attained for different Intel MPI versions (Tables 1-2) suggest a notable generational improvement. When comparing clock times, a reduction by up to 20% can be noted for Intel MPI version 2019. These savings, however, are accompanied by a significant increase in average memory – increase in RAM usage of up to 250% is observed when using Intel MPI version 2019.

Table 3: Scaling in OpenFOAM 8 when using OpenMPI version 4.1.1.

Cores	**32**	**64**	**96**	**128**	**256**	**512**	**768**	**1024**
Clock time [s]	63629	30001	19986	15857	9110	7056	7106	6989
Avg. time per TS [s]	1.150	0.545	0.366	0.299	0.170	0.126	0.124	0.120
RAM usage [GB]	7.123	9.471	11.849	14.208	23.823	45.321	68.624	94.411

OpenFOAM 8 with OpenMPI version 4.1.1 behaves similarly to the variant with Intel MPI version 2019 (Table 2). Although on average slightly slower in terms of overall computational time, as evidenced by the results presented in Table 3, this implementation is still adequate and viable option.

Table 4: Scaling in OpenFOAM v2012 when using OpenMPI version 4.1.1.

Cores	**32**	**64**	**96**	**128**	**256**	**512**	**768**	**1024**
Clock time [s]	57608	27312	17984	13490	7702	5289	5268	5251
Avg. time per TS [s]	0.999	0.490	0.328	0.247	0.148	0.092	0.092	0.091
RAM usage [GB]	7.321	9.866	12.411	14.998	25.592	48.057	71.138	96.805

Despite using the same OpenMPI version, OpenFOAM v2012 scales better than OpenFOAM 8 and provides reduction of up to 25% in computational time (clock time) when comparing equivalent test scenarios, as shown in Tables 3 and 4.

If we analyze the overall speedup, it is evident that the OpenFOAM v2012 with OpenMPI version 4.1.1 achieves the best results. Additionally, unlike in other test cases, there is no degradation in performance, although the efficiency is reduced and we are at the point of diminishing returns. Clearly, for a problem in question, 512 cores are the optimal choice, which suggest that approximately 7500 grid cells per core is a maximum for a given system and configuration. These results are in line with observations noted in [5]. With the subsequent increase in core count, the interconnect becomes a bottleneck. Speedup for conducted tests is given in Figure 4.

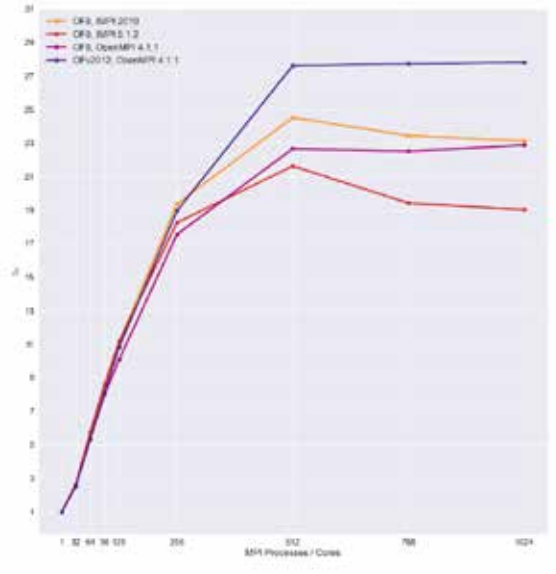


Figure 4: Speedup for different cases.

It is important to note that obtained results are influenced by hardware/software installed. Similar system configurations should achieve the same levels of scaling and speedup, particularly for a given problem type (incompressible, transient problem).

4. Conclusion

The main focus of this article is MPI associated computational speedup that can be achieved for a given problem. The problem in question, oil spill and its propagation as influenced by the sea currents, is an ecological problem that relies on appropriate response in a timely manner. For that purpose, data from the Regional Ocean Modelling System is utilised to simulate real-world conditions in Kvarner Bay and estimate the extents of the oil spill after a given period. Oil spill propagation is simulated as a simple scalar transport problem.

For the problem in question, computational efficiency is quintessential as it enables prompt propagation estimates and consequently quick response. With that in mind, two variants of the open-source CFD code OpenFOAM have been assessed with regards to MPI libraries at their disposal. This allows users to assess, and based on needs and hardware capabilities, tune their distributions in order to achieve optimal performance. We have clearly shown that there is a significant difference between OpenFOAM variants themselves as well as between the MPI libraries for a given option. Depending on the configuration, adoption of a proper MPI library can lead to a 25% reduction in computational time while generational advancements can additionally provide up to 25%.

5. References

1. *(scientific article, printed)* Lončar, G., Beg Paklar, G. & Janeković, I. (2012) Numerical modelling of oil spills in the area of Kvarner and Rijeka bay (The northern Adriatic Sea). *Journal of Applied Mathematics.* 2012 (497936), 1-20.
2. *(scientific article, printed)* Ivić, S., Mrša Haber, I. & Legović, T. (2017) Lagrangian coherent structures in the Rijeka bay current field. *Acta Adriatica.* 58 (3), 373-390.
3. *(scientific article, printed)* Mrša Haber, I., Legović, T., Kranjčević, L. & Cukrov, M. (2020) Simulation of pollutants spreading from a sewage outfall in the Rijeka bay. *Mediterranean Marine Science.* 21 (1), 116-128.
4. *(report)* Culpo, M. (2011) Current bottlenecks in the scalability of OpenFOAM on massively parallel clusters. *PRACE.* PRACE white paper 2011.
5. *(book chapter)* Axtmann, G. & Rist, U. (2016) Scalability of OpenFOAM with large eddy simulations and DNS on high-performance systems. In: Nagel, W. E., Kröner, D. H. & Resch, M. M. (eds) *High Performance Computing in Science and Engineering '16: Transactions of the High Performance Computing Center, Stuttgart (HLRS) 2016.* Cham, Springer, pp. 413-424.
6. *(report)* Keough, S. (2014) Optimising the parallelisation of OpenFOAM simulations. *Defence Science and Technology Organisation.* Report Number: DSTO-TR-2987.
7. *(scientific article, printed)* Shchepetkin, A. F. & McWilliams, J. C. (2005) The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean modelling.* 9 (4), 347-404.
8. *(software)* OpenFOAM 8. (2020) London. Weller, H.
9. *(software)* OpenFOAM v2012. (2020) Bracknell. OpenCFD Ltd.
10. *(scientific article, printed)* Menter, F.R. (1994) Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal.* 32 (8), 1598-1605.