Published in proceedings of IEEE International Conference on High Performance Computing (HiPC 2015), 16-19 December 2015, Bengaluru, India, which should be cited to refer to this work.

Multiscale multiphysics process on a HPC infrastructure: Application to coral growth process

Karthik Senthil*
Dept. of Information Technology
National Institute of Technology
Karnataka, India
karthik.senthil94@gmail.com

Paul Albuquerque
inIT institute
University of Applied Sciences
and Arts Western Switzerland
Geneva, Switzerland
paul.albuquerque@hesge.ch

Jonas Latt
Centre Universitaire Informatique
University of Geneva
Geneva, Switzerland
jonas.latt@unige.ch

***Abstract*—Many natural processes are characteristic multiscale multiphysics problems. Over the years techniques have been developed to study and simulate these processes using a computer. Such simulations are highly resource intensive and their performance is computationally bound on a large scale. High Performance Computing (HPC) plays a handy role in such a scenario. However, deploying a multiscale multiphysics application on a HPC infrastructure requires identifying and further tuning some parameters so as to improve performance and efficiency. This paper explains this procedure for the case study of a nutriment-driven coral growth process.**

***Index Terms*—Scientific & High Performance Computing, Lattice Boltzmann Method, Multiscale Multiphysics Science, Performance Analysis**

I. Introduction

Corals are marine invertebrates that live in tropical oceans. They are classified as filter-feeding marine sessile organisms, *i.e.* biological systems in which growth and development process is affected by the suspended particle distribution (*e.g.* nutriments) in their environment.

This paper explains a simulation model for the growth of a coral from a multi-scale, massively parallel perspective. It is designed as a tool for biologists. Moreover, it provides a basis for the determination of optimal parameters, in view of proper scaling and efficient performance for this model on a High Performance Computing (HPC) infrastructure. The model is fully three-dimensional (3D) and was developed with the open-source library Palabos[1], based on the Lattice-Boltzmann method. The model was deployed and tested on the supercomputer Lemanicus BlueGene/Q.

A. Physics of the coral growth process

The coral growth process is an ideal example of a multiscale multiphysics problem [1]. In the present work, we address the multiscale nature of the space and time scales in different ways. At the level of the time scales, the coral growth model is artificially accelerated to match the time scales of the fluid more closely, and yet produce the same physics. This model, described in [2], implements coral growth at discrete iterations, which are implemented whenever the fluid finds the time to adapt to the updated coral geometry and reaches a statistically stationary state. At the level of the spatial scales, we chose to represent the physics accurately by fully resolving all scales, with the help of a supercomputing architecture.

B. Need for high performance computing

The simulation model spans over various large scale parameters which are not practically addressable on a general purpose computing machine. At each iteration, the fluid state is updated, the particles in suspension transported, and the overall coral structure grown. As an example, for a simulation domain of size $2.95[m] \times 0.9[m] \times 0.9[m]$ and maximum growth time of $32[s]$ (at the real time scale of

[1]www.palabos.org, *last accessed on 22 September 2015*

the fluid), around 14000 iterations are necessary. When run on a single standard Intel processor core, a total runtime of over an hour is needed for achieving only a negligible coral growth. This calls for parallelization on a HPC infrastructure. We use a form of parallelism [6] based on domain decomposition. Performance analysis details for the model are charted out in section III and their results accordingly explained in section IV-B.

II. Numerical methodology

A. Lattice Boltzmann method

The Lattice Boltzmann Method (LBM) [3] is a modern and powerful approach in computational fluid dynamics. In the present work, we chose this method as the fluid flow can be easily parallelized using LBM. Furthermore, the method is suitable for flows around complex and dynamically changing geometries like corals. Finally, the method is naturally able to incorporate a Lagrangian particle model to simulate the advection and absorption of nutriments [4].

B. Coral growth model

Our simulation model for coral growth is fully 3D. The simulations were animated using Paraview[2], an open-source application for interactive, scientific visualization. The coral growth algorithm was developed and tested on a small simulation domain of $2.95[m] \times 0.9[m] \times 0.9[m]$.

The growth model has 3 main components: (1) the fluid flow around the coral, (2) nutriments or suspended particles in the fluid, and (3) the growing coral structure.

1) ***Fluid flow around the coral****:* We simulated the coral growth process using a fluid with a kinematic viscocity of $0.001[m^2/s]$ and an initial Reynolds number of 17.1. The fluid was assumed to be present throughout the simulation domain. Further, we apply the BGK (Bhatnagar-Gross-Krook) collision model for the lattice representing the fluid and the required boundary conditions were applied on the fluid to represent the physical constraints in the coral growth process.

2) ***Nutriments in the fluid around the coral****:* The nutriments which are carried by the fluid to the coral sites can be modeled as tagged particles. These particles are characterised as Lagrangian passive tracers. These particles are injected into the domain from the top-surface in each iteration of the simulation. Along with an injection domain, 2 absorption domains are defined i.e. regions to absorb particles. The 2 absorption domains in our model are along the bottom surface(sea-bed) and along the outlet of the domain.

3) ***Coral structure growth algorithm****:* Initially the coral structure (also called "seed") is represented as a 3D cross placed at the center of the domain on the bottom plane of the lattice (Fig. 1a).

Coral growth is executed in cyclic, discrete steps. Between two coral growth iterations, the fluid and Lagrangian particles execute `coralGrowthIters` iterations. Whenever a particle enters a fluid cell neighboring a cell occupied by the coral, it gets absorbed (it is removed from the simulation), and a counter for this cell is incremented in the matrix `numParticles`. At the end of the `coralGrowthIters` iterations, the matrix `numParticles` is evaluated. All elements of this matrix exceeding a threshold value `threshold` are reset to zero, and the corresponding fluid cell gets converted into a solid cell, which is part of the coral. For the simulations presented in this article, we used the values `coralGrowthIters=400` and `threshold=100`.

III. Performance scaling and measurements on HPC infrastructure

A performance analysis of this model was established on the Lemanicus BlueGene/Q supercomputer of the CADMOS[3] center in Switzerland, utilizing a maximum of 8192 CPU cores for parallel executions.

A. Methodology for performance measurement

In the field of LBM, it is common to characterize the code performance by a value called ***Sites updated per second (SuS)***. This value is the number of lattice cells for which the code is able to perform collision-streaming cycle (including the update of

[2]www.paraview.org, *last accessed on 22 September 2015*

[3]Center for ADvanced MOdeling Science, www.cadmos.org

particles present on the same site) during a wall-clock second:

$$SuS = \frac{nx \cdot ny \cdot nz \cdot totalIters}{totalTime}$$

where the lattice resolution is $nx \times ny \times nz$, $totalIters$ is the number of iterations for which the program was run, and $totalTime$ is the measured wall-clock time.

1) ***Experiment on domain size/resolution****:* The performance of the model was measured for 7 different domain resolutions: 80(1×), 120(1.5×), 160(2×), 240(3×), 320(4×),480(6×) and 640(8×). In each case the model was executed with only the fluid (no particles) and the simulation was run for a maximum of 1000 iterations. The model was deployed on 512 cores.

2) ***Experiment on the number of processors/cores deployed (strong scaling)****:* In this test series, the strong scaling of the parallel program was investigated by solving a constant-sized problem with a varying number of processors. The domain was resolved with 320 grid cells, the particles were injected at 1 particle/cell, the threshold value for coral growth was 100 particles, and all simulations were run for 500 iterations. The reference value was taken with $c_{\text{ref}} = 64$ nodes, using 1 core/node (64 cores in total). In the first test case, this was compared to 3 simulation runs using 64, 128, and 256 nodes with 4 cores/node in each case. In the second test case, the simulation was again run on 64, 128, and 256 nodes, but with 8 cores/node.

3) ***Scaling the number of cores with the domain size (weak scaling)****:* In the following test for the weak scaling behavior of the code, the total number of grid nodes scales linearly with the number of cores c, *i.e.* the resolution r goes like

$$r = r_{\text{ref}} \times \left(\frac{c}{c_{\text{ref}}}\right)^{1/3}$$

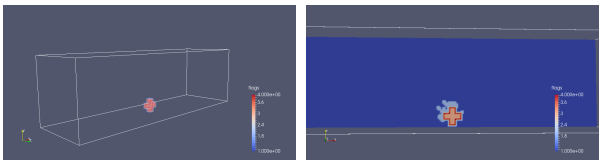
where $r_{\text{ref}} = 160$. In this experiment the threshold value for coral growth was 100 particles, the particles were injected at the rate of 1 particle/cell and simulations were run for 200 iterations. Here, we took $c = 64, 512, 1024, 2048, 4096$ and 8192 cores.

4) ***Experiment on particle/nutriment density****:* In this numerical experiment the performance of the model was evaluated for 3 different particle densities: 1 particle/cell, 2 particles/cell and 4 particles/cell. Again in this setup, the domain was resolved with 320 grid cells, 128 nodes were used (4 cores/node) and all simulations were run for a maximum of 500 iterations.

IV. Results

A. Coral growth algorithm

The coral growth algorithm and the model explained in section II-B were simulated on the BlueGene/Q supercomputer in order to showcase the growth of the coral and to measure its accuracy. In the simulation environment, the domain was chosen with size $2.95[m] \times 0.9[m] \times 0.9[m]$ and a resolution of 80 lattice units. The maximum simulation time was fixed at $300[s]$, the threshold for coral growth was 100 particles and up to 256 nodes of the supercomputer were used (4 cores/node). Fig. 1b displays a significant growth in the coral structure at the end of simulation.



(a) 3D initial view (b) 2D cut of grown coral

Fig. 1. Coral structure during the simulation

The growth rate and coral structure obtained are accurate and match the expected results.

B. Performance analysis on HPC infrastructure

We provide here the results obtained for the numerical experiments explained in section III-A

1) ***Numerical experiment on domain size/resolution****:* Fig. 2 shows a plot of performance measure (in terms of $[MSuS]$) vs. domain resolution. From these results we can infer that an increase in the total number of lattice sites in the domain increases the performance of the model implying that a greater extent of parallelism is achieved with a larger domain size. We also note that the increase in performance is not linear.

2) ***Numerical experiment on the number of cores****:* Table I summarizes the results. We can infer from them that the performance of the model increases with the number of cores used for the parallel execution. We also observe that the load on a node does not significantly increase with the number of cores used per node. Thus increasing the number of cores used per node does not drastically affect performance.

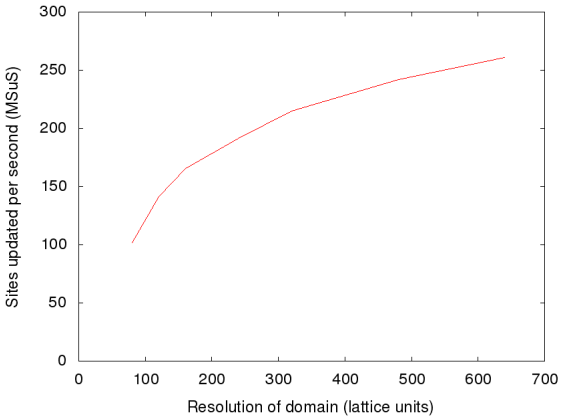


Fig. 2. Effect of domain resolution on performance

TABLE I
RESULTS OF NUMERICAL EXPERIMENT 2

Test case	Number of cores	Performance (in [MSuS])	Runtime (in [s])
Reference	64	15.9772	3334
4 cores per node	64 ×4 cores	45.8589	1161.65
	128 ×4 cores	77.6638	685.929
	256 ×4 cores	137.188	388.314
8 cores per node	64 ×8 cores	76.0773	700.234
	128 ×8 cores	135.278	393.797
	256 ×8 cores	205.613	259.088

*3) **Numerical experiment on changing the number of cores and the domain resolution**:* We observe that the performance of the model drastically increases with increase in the number of cores deployed for parallel execution along with a proportionate increase in the domain resolution. Despite the increase in total number of lattice sites for processing in the domain the increase in number of cores accounts for a greater extent of parallelism. Figure 3 provides a plot of performance of model(normalized with respect to reference test case) against the number of cores used. From this graph we observe that the performance improvement is almost linear and nearly consistent with the ideal scenario of linear performance gain.

*4) **Numerical experiment on particle density**:* The results of this numerical experiment have been tabulated in Table II. We can infer from these results that the performance of the model decreases while increasing particle (or nutriment) density. This is expected since more particles implies more processing to change physical parameters throughout the simulation domain.

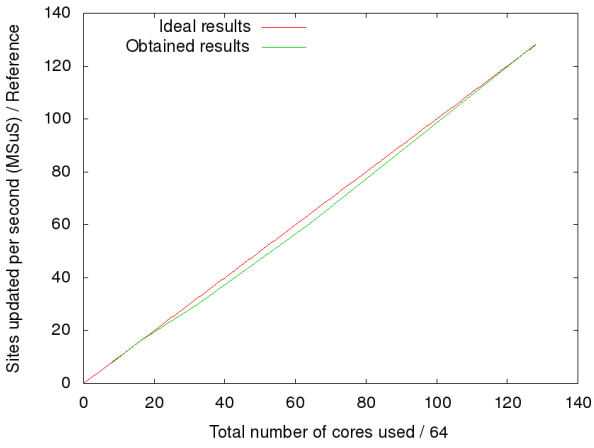


Fig. 3. Effect of changing simultaneously the number of cores and the domain resolution on the performance of the model

TABLE II
RESULTS OF NUMERICAL EXPERIMENT 4

Particle density (particles per cell)	Performance ($[MSuS]$)	Runtime ($[s]$)
1	77.0564	691.336
2	48.6513	1094.97
4	27.3364	1948.75

V. DISCUSSION ON SPEEDUP AND EFFICIENCY

We discuss here speedup and efficiency [5] for the proposed model. For a given resolution r, we assume the number of lattice sites/cells scales as $n_{\text{ref}} \cdot r^3$, where n_{ref} is the domain size in SI units. Sequential and parallel time complexities, T_{seq} and T_{par}, then follow the law

$$T_{\text{seq}} = c_1 \cdot n_{\text{ref}} \cdot r^3, \quad T_{\text{par}} = \frac{c_1 \cdot n_{\text{ref}} \cdot r^3}{p} + T_o(r,p)$$

where the constant c_1 is the time for a processor to update 1 cell, and $T_o(r,p)$ is the interprocessor communication time overhead.

A. Interprocessor communication time overhead

For a parallel simulation, the overhead time due to interprocessor communication must be accounted

for. Here we consider the the domain to be split vertically across the length. The overhead is thus proportional to the area of the split surface

$$T_o(r,p) = c_2 \cdot n_{\text{ref}} \cdot r^2$$

where the constant c_2 depends on the communication bandwidth. Hence,

$$T_{\text{par}} = \frac{c_1 \cdot n_{\text{ref}} \cdot r^3}{p} + c_2 \cdot n_{\text{ref}} \cdot r^2$$

B. Speedup, efficiency and iso-efficiency

Based on the expressions for T_{seq} and T_{par}, we can derive the speedup S and efficiency E formulae

$$S = \frac{T_{\text{seq}}}{T_{\text{par}}} = \frac{1}{\frac{1}{p} + \frac{c_3}{r}}$$

and, setting $c_3 = \frac{c_2}{c_1}$, we get $E = \frac{S}{p} = \frac{1}{1 + c_3 \cdot \frac{p}{r}}$

The iso-efficiency function expresses the dependency between r and p so as to maintain E constant. Thus, it is of the form $r = k \cdot p$ for some constant k.

C. Analysis on parallel time complexity

We also further analyzed T_{par} for a fixed domain resolution $r = 320[\text{lattice units}]$ by writing

$$T_{\text{par}} = \alpha \cdot \frac{1}{p} + \beta$$

and applying Least Square Interpolation (LSI). Fig. 4 plots T_{par} vs. the number of processors p. The fit for T_{par} using the LSI values $\alpha = 2.2 \times 10^5$ and $\beta = 200$ is shown together with the experimental results. This seems to confirm the correctness of our expression for T_{par}.

VI. Conclusion

In this paper we developed a new simulation model for showcasing a multiscale multiphysics process, a coral growth process, on a HPC infrastructure. We also investigated its efficiency and performance. As a matter-of-fact this work could be used by biologists as a tool or a standard use-case to determine the right factors for simulating such a problem and fine tune the parameters so as to produce an efficient and realistic simulation.

From the results, we conclude that the implemented model shows the expected growth rate and growth pattern of a coral. Similarly, the numerical

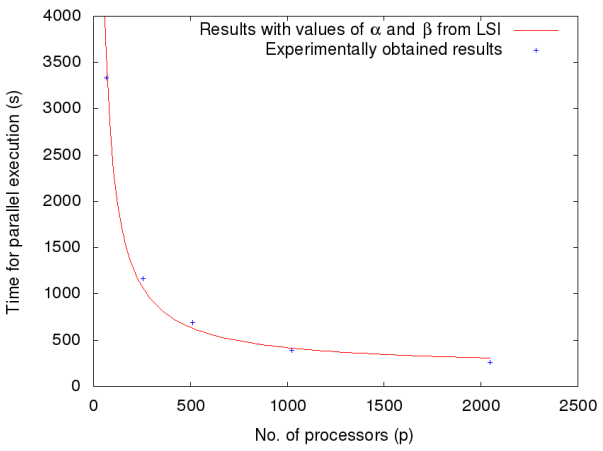


Fig. 4. Parallel execution time T_{par} vs. number of processors p

experiments help us identify the right parameters to analyze and improve the parallel performance of the model. The speedup and efficiency analysis also showcase the correctness of the methodology. We also concur that the simulation library Palabos is very robust and suitable for designing multiscale multiphysics problems.

Acknowledgment

We wish to acknowledge the CADMOS center for sponsoring part of this research, and for providing computation time on the BlueGene/Q supercomputer.

References

[1] J. Borgdorf et al. Foundations of Distributed Multiscale Computing: Formalization, Specification, Analysis and Execution. *J. Par. & Distr. Comp.* 73(4):465-483, 2013.

[2] Jaap A. Kaandorp et al. Effect of Nutrient Diffusion and Flow on Coral Morphology. *Phys. Rev. Lett.* 77(11):2328-2331, 1996.

[3] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems.* Cambridge University Press, 1998.

[4] A.J.C. Ladd and R. Verberg. Lattice-Boltzmann simulations of particle-fluid suspensions. *J. Stat. Phys.* 104(5):1191-1251, 2001.

[5] A. Grama et al. *Introduction to Parallel Computing.* 2nd edition, Pearson Education, 2003.

[6] B. Chopard et al. A Lattice Boltzmann simulation of the Rhone river. *Intl. J. Modern Phys.* 1340008, 2013