Seminar Report

GPU-Accelerated Simulation of Phytoplankton Convection Using Oceananigans.jl in Julia

Davide Mattioli

MatrNr: 28350066

Supervisor: Tino Meisel

Georg-August-Universität Göttingen
Institute of Computer Science

September 30, 2024

Abstract

This work aims to explore fluid dynamics simulation in Julia, focusing on an application of GPU computing on a simulation regarding the mixing of phytoplankton by convection in the ocean [TF11]. Fluid dynamics models used in oceanography and atmospheric sciences are computationally intensive, as such GPUs have increasingly replaced CPUs as the main computing source in this field due to their ability to parallelize complex calculations. Existing ocean simulation libraries were written for CPU architectures facing limitations in both speed and scalability when adapted to GPU-based systems.

To address this, the `Oceananigans.jl` library was developed as a GPU-optimized tool boasting the Julia's high-level syntax and Just-In-Time (JIT) compilation to achieve both computational efficiency and flexibility. This report examines the application of `Oceananigans.jl` to a C-Grid [LCL10] ocean circulation of plankton convection by simulating the interaction and fluid body movement happening in the ocean. It will also evaluate the library's performance on different grid sizes [Sil+23].

The proposed solution demonstrates how `Oceananigans.jl` efficiently simulates complex oceanographic processes using highly customizable grids. making it suitable for large-scale fluid simulations using Boussinesq equations [Kar12] and WENO schemes [Sil+24]. Performance benchmarks indicate a significant computational advantage of GPU-based simulations over CPU-based ones, particularly for large grid sizes. The results confirm that `Oceananigans.jl` provides an effective and scalable tool for modern fluid dynamics simulations in scientific computing.

Declaration on the use of ChatGPT and comparable tools in the context of examinations

In this work I have used ChatGPT or another AI as follows:

- ☐ Not at all
- ☑ During brainstorming
- ☐ When creating the outline
- ☑ To write individual passages, altogether to the extent of 5% of the entire text
- ☐ For the development of software source texts
- ☐ For optimizing or restructuring software source texts
- ☑ For proofreading or optimizing
- ☐ Further, namely: -

I hereby declare that I have stated all uses completely.
Missing or incorrect information will be considered as an attempt to cheat.

Contents

List of Tables

List of Figures

List of Listings

List of Abbreviations

WENO	The Weighted Essentially Non-Oscillatory
GPU	Graphical processing unit
CPU	Central processing unit
CFL	Courant–Friedrichs–Lewy
CUDA	Compute Unified Device Architecture

1 Introduction

In fluid dynamics the Boussinesq equations are used to model water waves, this is because they can account for non-linearity, when waves interact and change shape in complex ways, and frequency dispersion, when waves of different lengths travel at different speeds, causing them to spread out over time. Simpler models like the shallow water equations cannot handle such complexity, this reduces their accuracy in representing real-world wave behaviours, particularly in more dynamic or variable conditions, as they oversimplify wave interactions and treat all wave lengths as travelling at the same speed. They're used in coastal engineering and oceanography by simulating long waves and their interactions with underwater landscapes, including wave refraction and shoaling [Fan+22].

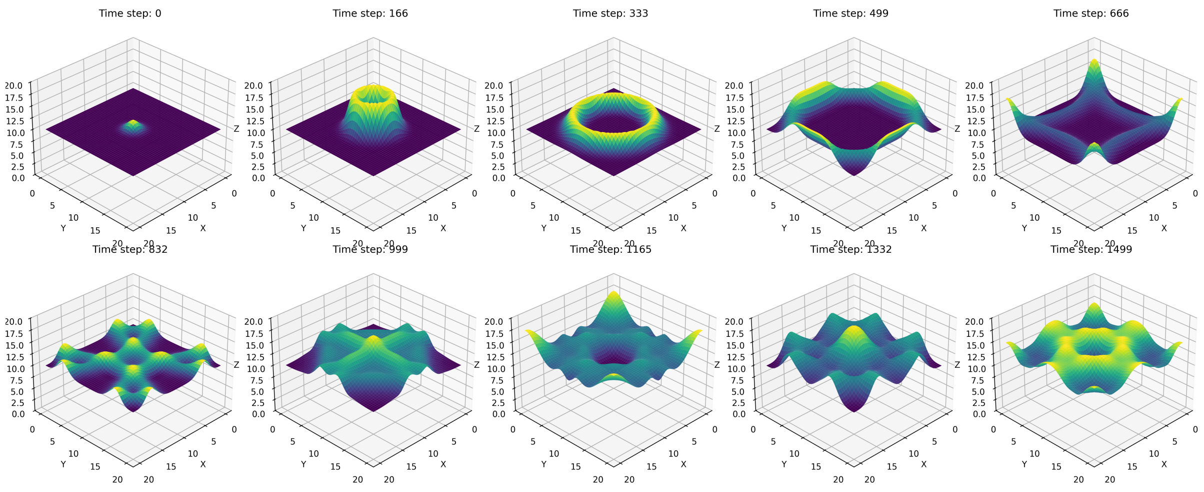

Figure 1: Wave simulation with Boussinesq equations

This report will have a look at their application in ocean circulation models where the Boussinesq equations can be used to simplify the differential equations by assuming density is constant, except in the vertical buoyancy forces, which helps model ocean currents.

These equations are computationally heavy due to their non-linear nature and the need to account for wave frequency dispersion.Oceananigans.jl, an open-source library written in Julia, unlike traditional ocean models that are ported from CPU-based code, was built from scratch to run efficiently on GPUs. It leverages Julia's high-level programming features and Just-In-Time (JIT) compilation to achieve performance similar to C or Fortran, while allowing for flexibility and ease of use [Sil+23].

2 Oceananigans Hydrostatic Model

Oceananigans.HydrostaticFreeSurfaceModel solves the hydrostatic Boussinesq equations using the finite volume method, which by splitting the ocean into small grid cells makes it able to be computed more efficiently by the GPU. It uses kernel fusion to optimizes performance on GPUs by combining large parts of the calculation into a single, compute-heavy kernel, reducing memory usage and boosting efficiency. This makes Oceananigans highly memory-efficient, enabling global ocean simulations at high resolutions on a single Nvidia V100 GPU, while also providing performance portability across CPUs and GPUs.

2.1 C-grid division

This division is called C-grid, which uses a layout that helps accurately compute fluid flow by storing different types of information (like velocity and pressure) at different points in the grid cells, this method is mainly used in weather forecasting by dividing the territory in small volume units. [Sil+24]

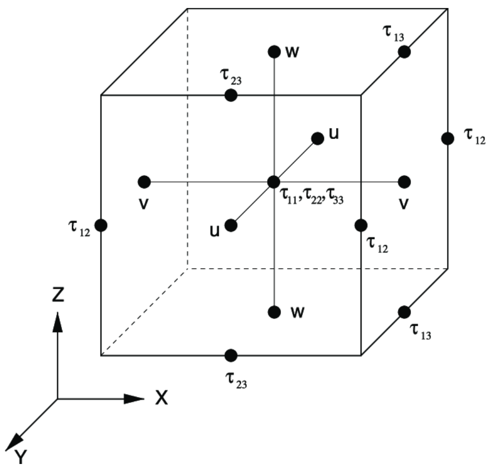

Figure 2: Example of a C grid from [LCL10]

2.2 Adams-Bashforth time-stepping algorithm

The model uses a second-order Adams-Bashforth time-stepping algorithm to simulate water movement over time by incorporating both the current and previous state of the system. This approach is effective in handling the coexistence of low-frequency advection and high-frequency wave propagation found in oceanic and atmospheric flows. To ensure computational stability the Courant–Friedrichs–Lewy (CFL) condition is applied by constraining the time step based on the speed of the waves or fluid flow being simulated and the resolution of the grid (the size of each grid cell). If the time step is too large and exceeds this threshold, the simulation can become unstable, producing inaccurate or nonsensical results[Kar12]. So with the Adams-Bashforth scheme we can have larger and stable time steps without violating this condition. This ensures that the model remains stable even in the presence of rapid wave motions while accurately capturing slower processes like advection.

2.3 Weighted Essentially Non-Oscillatory (WENO) Scheme in Oceananigans

The Weighted Essentially Non-Oscillatory (WENO) is a high-order numerical method that compute hyperbolic partial differential equations, it is made to solve those that involve sharp gradients or discontinuities like shock waves or steep fluid fronts. WENO uses a weighted combination of several polynomial reconstructions from neighbouring points to approximate the solution at each point. These weights are then dynamically adjusted based on the smoothness of the solution, with more accurate reconstructions higher-order in smooth regions and more stable reconstructions near discontinuities[Sil+24].

In Oceananigans it is used to compute momentum advection, as ocean simulations often involve sharp features like ocean currents where fluid properties such as velocity, temperature, or salinity can change over very short distances. These values cannot be accurately computed without introducing artificial oscillations that can arise from the steep gradients. Traditional methods like lower-order upwind schemes are more dissipative and smooth out these sharp transitions, reducing the accuracy of the simulation[Sil+24] .

3 Plankton Convection

Phytoplankton are the primary oxygen producers in marine ecosystems and represent a big portion of the total biomass in the ocean, to analyse their movement in the water we can apply the same concepts from oceanic convection which drive the mixing and distribution of these organisms in the Oceanic water. The phenomenon that drives this motions and that will be analysed is called spring phytoplankton bloom, a steady increase in phytoplankton biomass after periods of convective mixing. This process is generally explained by the "critical turbulence hypothesis," which suggests that as turbulent mixing subsides, phytoplankton remain in the upper, sunlit layers of the ocean, enabling growth due to an increased light exposure. In this Report, we simulate the convective mixing of phytoplankton using the `Oceananigans.jl` library, replicating the conditions that lead to a phytoplankton bloom as mixing gradually diminishes. The simulation is designed to reflect the work of Taylor and Ferrari (2011) [TF11], which showed how the weakening of convection triggers phytoplankton blooms.

The model simulated several processes that influence phytoplankton concentration dynamic:

- **Advection**: Transport of phytoplankton by the flow of water.
- **Diffusion**: The spreading of phytoplankton due to random motion.
- **Growth**: Phytoplankton reproduction, dependent on sunlight availability, typically higher near the surface.
- **Mortality**: The loss of phytoplankton biomass due to predation by zooplankton and viral decay.

These processes of diffusion are governed by the following equation for the concentration of phytoplankton P:

$$\partial_t P + \mathbf{v} \cdot \nabla P - \kappa \nabla^2 P = \left[\mu_0 \exp\left(\frac{z}{\lambda}\right) - m\right] P$$

where:

- $\mathbf{u}$ represents the velocity field of the water,
- κ is the diffusivity,
- μ_0 is the growth rate of phytoplankton at the surface,
- λ is the attenuation scale of sunlight with depth, and

- m is the mortality rate.

The time-dependent surface *buoyancy flux*, which drives the convective mixing, is also included in the model. As the simulation progresses, this flux decays, showing a seasonal reduction in turbulence that corresponds to the transition from mixed to stratified conditions.

3.1 Grid Setup

```julia
grid = RectilinearGrid(GPU(),
                       size=(128, 128),
                       extent=(64, 64),
                       halo=(3, 3),
                       topology=(Periodic, Flat, Bounded))

```

Listing 1: Grid settings

The simulation uses a two-dimensional rectilinear grid defined with 128^2 points and 3^2 halo points to enable high-order advection schemes, such as the previously mentioned WENO. The grid spacing in both x and z directions is set to 1 meter by default to capture small-scale processes while maintaining computational efficiency. The grid is periodic in the x-direction to ensure continuity at the boundaries, the opposite is true for z-direction is bounded, simulating the ocean surface and seafloor. The periodic boundary in the x-direction and the bounded condition in the z-direction reflect real oceanic environments, where currents loop around, but vertical mixing is constrained by natural barriers.

3.2 Buoyancy Flux

In the context of plankton convection, the surface buoyancy flux is the parameter that regulates the intensity of convective turbulence in the ocean.For instance, during winter the ocean experiences strong cooling, leading to negative buoyancy fluxes and intense turbulent mixing distributing nutrients and removing phytoplankton from the upper, sunlit layers of the ocean, preventing a bloom. The reduction in buoyancy flux, leads to weaker turbulence allowing phytoplankton to remain in the euphotic zone, where they can receive enough sunlight for photosynthesis, thus initiating a spring bloom [TF11]. So, in the model we define the surface buoyancy flux as a time-dependent variable with an exponential decay with time described by the following equation:

$$\text{buoyancy\_flux}(t) = \text{params.initial\_buoyancy\_flux} \times \exp\left(-\frac{t^4}{24 \times \text{params.shut\_off\_time}^4}\right)$$

```julia
buoyancy_flux_parameters = (initial_buoyancy_flux = 1e-8,
                            shut_off_time = 2hours)
```

The parameters used are an initial buoyancy flux of $1 \times 10^{-8}\,\mathrm{m}^2\,\mathrm{s}^{-3}$ and a shut-off time of 2 hours. This is a scenario where the surface buoyancy flux rapidly diminishes, causing the subsidence of turbulent surface mixing as cooling weakens.

To capture the reduction of surface cooling over time we fix a boundary condition for the buoyancy flux at the surface of the ocean model using a *FluxBoundaryCondition*, allowing the simulation to capture the reduction of surface cooling over time. As the buoyancy flux approaches zero, turbulent mixing in the simulation diminishes, simulating the transition from winter to spring conditions where convective mixing ceases and stratification begins.

The boundary condition is implemented as follows:

```

buoyancy_flux_bc = FluxBoundaryCondition(buoyancy_flux,
                    parameters = buoyancy_flux_parameters)
```

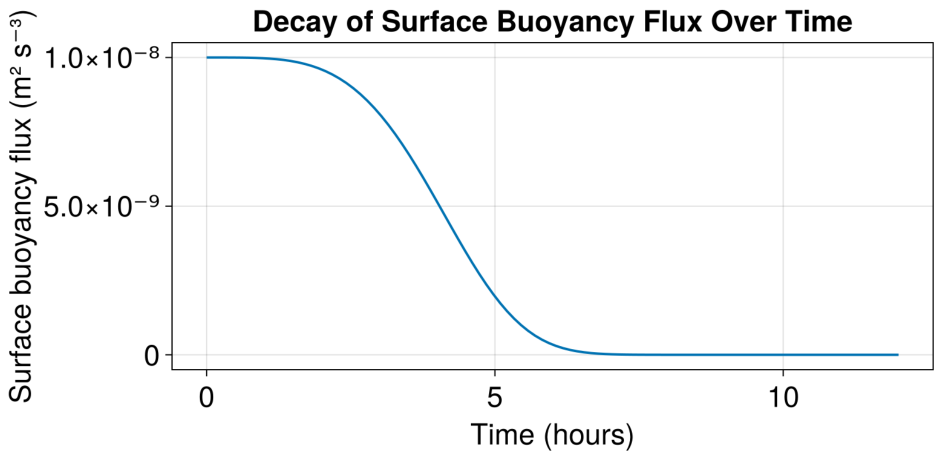

Figure 3: Decay of Buoyancy flux over time

As shown in the Figure 3, during the first few hours, the buoyancy flux is relatively high, reflecting strong mixing. However, around the 3-hour mark, the surface buoyancy flux drops sharply, indicating a reduction in turbulent mixing. The nearly flat section of the curve after hour 7 represents a state where the buoyancy flux has decayed to zero, indicating that convective turbulence has "shut down" triggering the bloom according to the authors' hypothesis. The research done by Taylor [TF11] in particular emphasizes that the reduction in surface buoyancy flux provides a better indicator for the onset of phytoplankton blooms than the mixed-layer depth alone.

3.3 Plankton Growth

To model the plankton population dynamics, we introduce a function that computes net growth of phytoplankton is defined by the biological growth and mortality. The growth rate, μ_0, represents the surface growth rate of phytoplankton under ideal conditions where light is plentiful. However, as depth increases, the availability of light diminishes exponentially, so a decay factor, λ, will model how phytoplankton's access to sunlight decreases with depth. The mortality rate m, will balance the population to a reasonable amount by taking into account viral infections and grazing by zooplankton.

The function for net phytoplankton growth is represented as:

$$\frac{\partial P}{\partial t} = \left(\mu_0 \cdot \exp\left(-\frac{z}{\lambda}\right) - m\right) \cdot P$$

Where P is the phytoplankton concentration at depth z. The net rate of change in phytoplankton concentration depends on the balance between growth and mortality. At shallow depths, where sunlight is abundant, growth dominates, but as depth increases, growth slows and mortality becomes more significant.

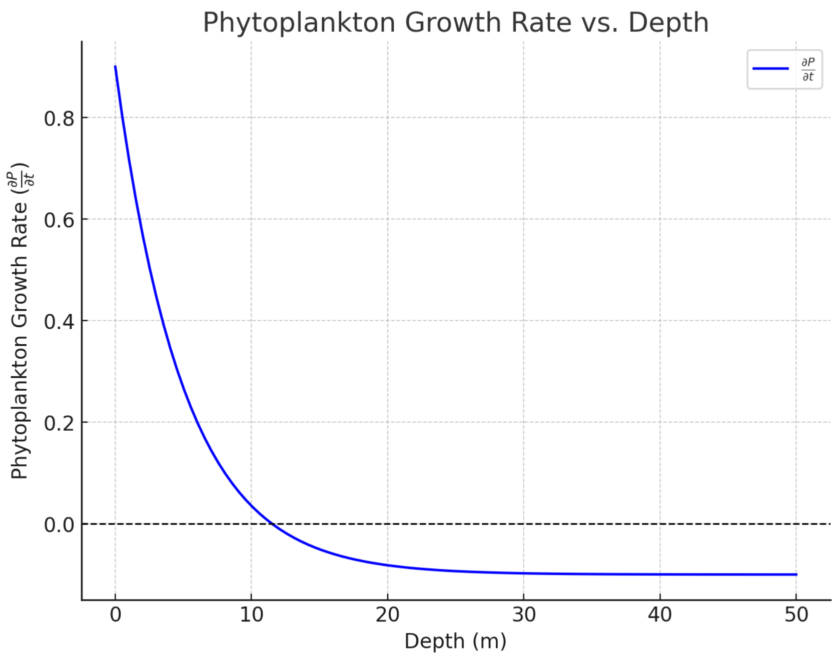

Figure 4: Phytoplankton growth rate at different depth

3.4 Model Overview

The final model of the simulation is then based on all the characteristics described above, in essence: a rectilinear grid with periodic boundary conditions in the horizontal direction, and bounded conditions in the vertical direction. The non-hydrostatic assumption holds in this simulation to account for vertical accelerations of the biomass movement. The model will also include a momentum advection using a fifth-order WENO scheme; this ensures that sharp gradients in velocity fields are resolved without introducing numerical oscillations. The goal of this model is to simulate the dynamics of the phytoplankton concentration and of the buoyancy.

The specific configuration of the `NonhydrostaticModel` used in this study is as follows:

```
model = NonhydrostaticModel(; grid,
                            advection = WENO(),
                            timestepper = :RungeKutta3,
                            closure = ScalarDiffusivity(=1e-4, =1e-4),
                            coriolis = FPlane(f=1e-4),
                            tracers = (:b, :P),
                            buoyancy = BuoyancyTracer(),
                            forcing = (; P=plankton_dynamics),
                            boundary_conditions = (; b=buoyancy_bcs))
```

- **Grid:** We pass the 128×1×128 rectilinear grid in the GPU
- **Advection Scheme:** The advection of momentum is computed using a 5th-order WENO.
- **Time Stepping:** The Runge-Kutta time-stepping function is used to approximate solutions to ordinary differential equations (ODEs).A third-order Runge-Kutta specifically refers to methods that achieve third-order accuracy having a local truncation error per time step is proportional to h^4, and the global error over an interval proportional to h^3, where h is the time step size.
- **Tracers:** Two tracers are included in the model, B for buoyancy and P for phytoplankton concentration.
- **Closure:** A scalar diffusivity model is used for closure, it specifies the rate of phytoplankton diffusion in a fluid, due to molecular motion or turbulence. with diffusivities set to $\nu = 0.0001\,\mathrm{m^2\,s^{-1}}$ for momentum and $\kappa = 0.0001\,\mathrm{m^2\,s^{-1}}$ for the tracers, representing small-scale turbulent mixing.
- **Buoyancy and Coriolis:** The model incorporates a buoyancy tracer to simulate the effects of buoyant forcing in the vertical direction, and an ff-plane approximation is used to model the Coriolis effect, which is important in large-scale geophysical flows.

3.5 Initial Conditions

The initial conditions are defined to try to replicate a stratified ocean column. The upper layer of the ocean, where water is relatively uniform in temperature, salinity, and density, is called mixed-layer, and it is set to be at 32 meters of depth in order to separate the well-mixed surface waters from the stratified layers below. This stratification is needed to check how buoyancy varies with depth, as it will influence the vertical distribution of energy and matter in the ocean.

The stratification profile function operates in this way: if the depth z is below the mixed layer depth, the buoyancy increases Nlinearly with depth, otherwise it will remain constant with a value of N^2 times the maximum depth:

$$\text{stratification}(z) = \begin{cases} N^2 \cdot z & \text{if } z < -\text{mixed\_layer\_depth} \\ -N^2 \cdot \text{mixed\_layer\_depth} & \text{otherwise} \end{cases}$$

To make the simulation more realistic, noise is added to the stratification. These perturbations will simulate the inhomogeneous environment present in the ocean and it is computed as:

$$\text{noise}(z) = 1 \times 10^{-4} \cdot N^2 \cdot \text{grid.Lz} \cdot \text{randn}() \cdot e^{z/4}$$

This will also induct random fluctuations in the buoyancy, ensuring an initial state not perfectly smooth.

The initial buoyancy, which combines both the stratification and the noise, is set by:

$$\text{initial\_buoyancy}(x, z) = \text{stratification}(z) + \text{noise}(z)$$

Lastly, the concentration of phytoplankton (P) is initialized with a uniform distribution, uniformly with a value of 1 across the grid.

```julia
set!(model, b = initial_buoyancy, P = 1)
```

3.6 Results

The results are from a simulation of 24 hours and the output variable are vertical velocity and plankton "concentration" as a heatmap. The buoyancy flux over time is also and the plankton concentration over different level of different z are also diplayed.

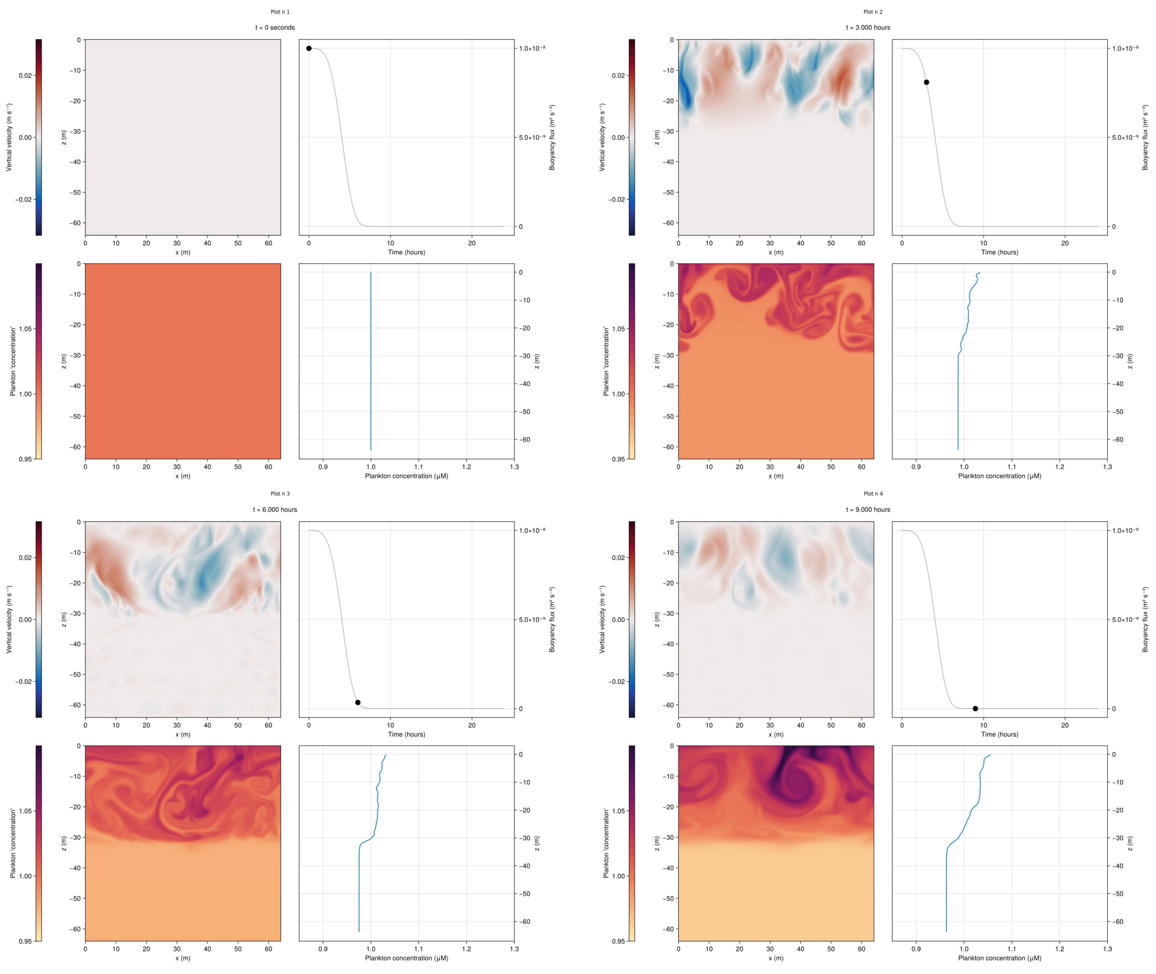

Figure 5: first 4 simulation time snapshots

In the first frame (t = 0 seconds), we see a calm ocean with no turbulence, this uniform phytoplankton concentration represent the early winter stage where strong atmospheric cooling prevents a bloom by continuously mixing plankton out of the Euphotic zone, where the bloom is common to happen as more light pass through the water. In the second frame (t = 3 hours) turbulence starts to develop, small eddies redistribute phytoplankton on the vertical axis (z). However, as buoyancy flux decreases, surface cooling weakens, setting the stage for a shutdown of turbulent mixing.

By the third frame (t = 6 hours) and the fourth frame (t = 9 hours), turbulent mixing has intensified, but the weakening buoyancy flux signals the eventual subsidence of convection. According to Taylor and Ferrari [TF11], blooms are still unlikely at this stage because strong turbulence prevents phytoplankton from remaining near the surface.

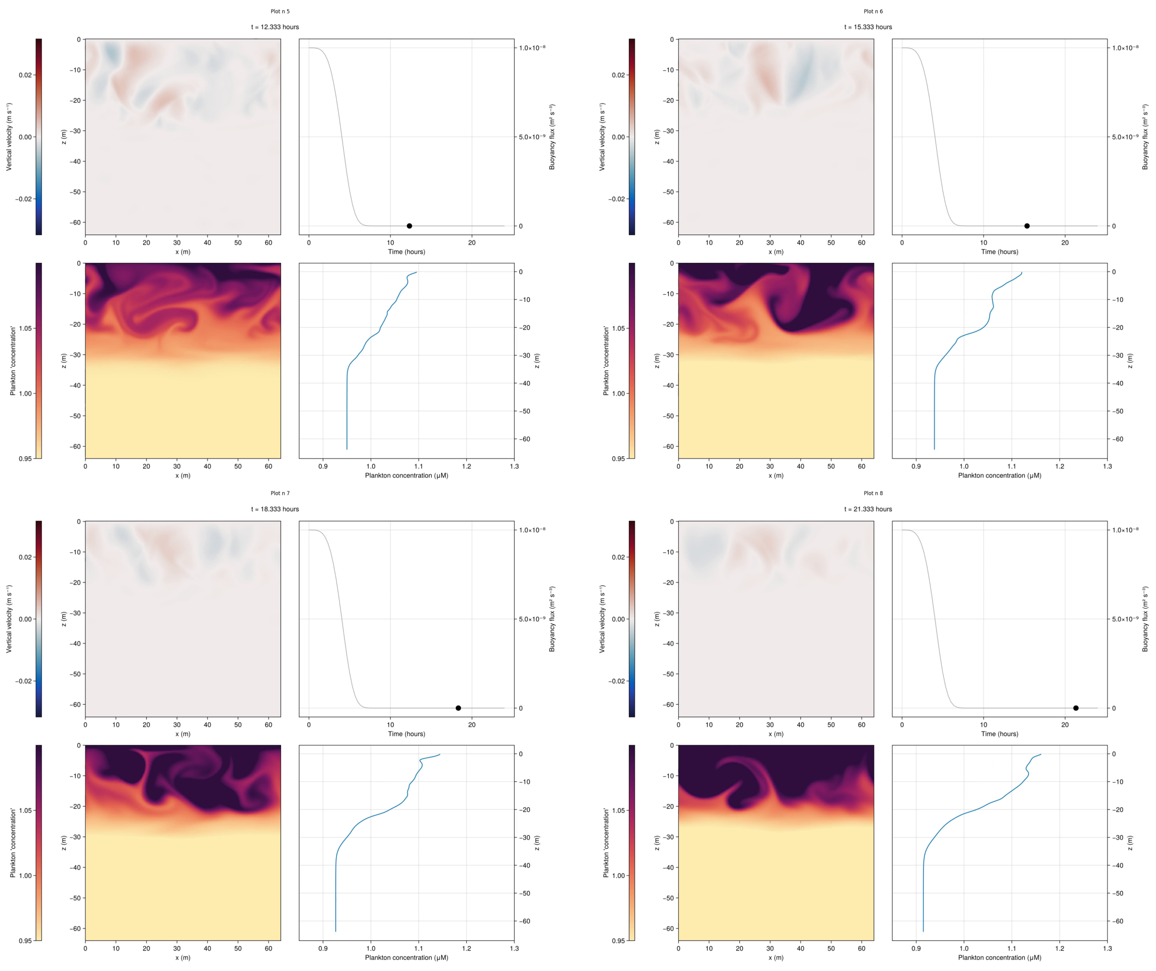

Figure 6: last 4 simulation time snapshots

The progression of frames from 5 to 8 shows the gradual shutdown of turbulent convection and its direct impact on phytoplankton concentration, aligning with Taylor and Ferrari's hypothesis on bloom initiation. The buoyancy flux steadily declines causing a slow-down of the velocity, this allows the phytoplankton to accumulate near the surface. In the earlier frames (5 and 6), small eddies still persist, but not enough to push the phytoplankton downward, this leads to a more stratified concentration in the euphotic zone. In the last two frames (7 and 8), the turbulence has essentially ceased marking the final stage of bloom initiation, where phytoplankton will stay in the upper layers proliferating from the constant exposure from sunlight and thus resulting in a environmental stagnation.

4 Benchmarks

Several benchmarks of the simulation have been computed on a machine with:

- CPU: Intel Core i7-13700HX, 8 cores, 2.1 GHz with 3.1Ghz as Boosting
- GPU: RTX 4070 8 GB VRAM 15.6 TFLOPS FP16, 15.6 TFLOPS FP32
- RAM: 32 GB DDR5 6400 Mhz

Table 1: CPU Simulation Results

Grid Size	Initialization Time (ms)	Initial Step Time (s)	Total Simulation Time (s)
16x16	95.893	8.406	10.668
32x32	124.825	8.659	11.550
64x64	79.410	4.659	10.464
128x128	119.257	5.575	25.143
256x256	212.983	8.641	108.300
512x512	211.162	5.382	378.660
1024x1024	690.176	7.909	1583.580
2048x2048	NaN	NaN	NaN
4096x4096	NaN	NaN	NaN

Table 2: GPU Simulation Results

Grid Size	Initialization Time (ms)	Initial Step Time (s)	Total Simulation Time (s)
16x16	1.211	5.066	7.599
32x32	1.016	5.070	8.001
64x64	1.098	5.019	8.226
128x128	1.794	4.896	7.663
256x256	4.922	4.610	8.600
512x512	11.405	4.757	12.623
1024x1024	19.833	4.458	30.980
2048x2048	71.825	228.210	112.860
4096x4096	166.034	5.099	463.440

For smaller grid sizes (16x16 to 128x128), both the CPU and GPU perform similarly in terms of total simulation time. However, with bigger grid size from 256 onwards the GPU outperform the CPU by a large margin. For instance, on a 1024x1024 grid, the GPU completes the simulation in approximately 31 seconds, whereas the CPU takes over 26 minutes (1583.58 seconds).

The implementation of CUDA kernels in Oceananigans make the machine to fully leverage the parallel processing power of the GPU, allowing for faster computation and handling of large-scale simulation. This is explained by the fact that differential equations employed in the simulation really benefit from the parallelisation of the computation, allowing a better performance on hardware with more cores.

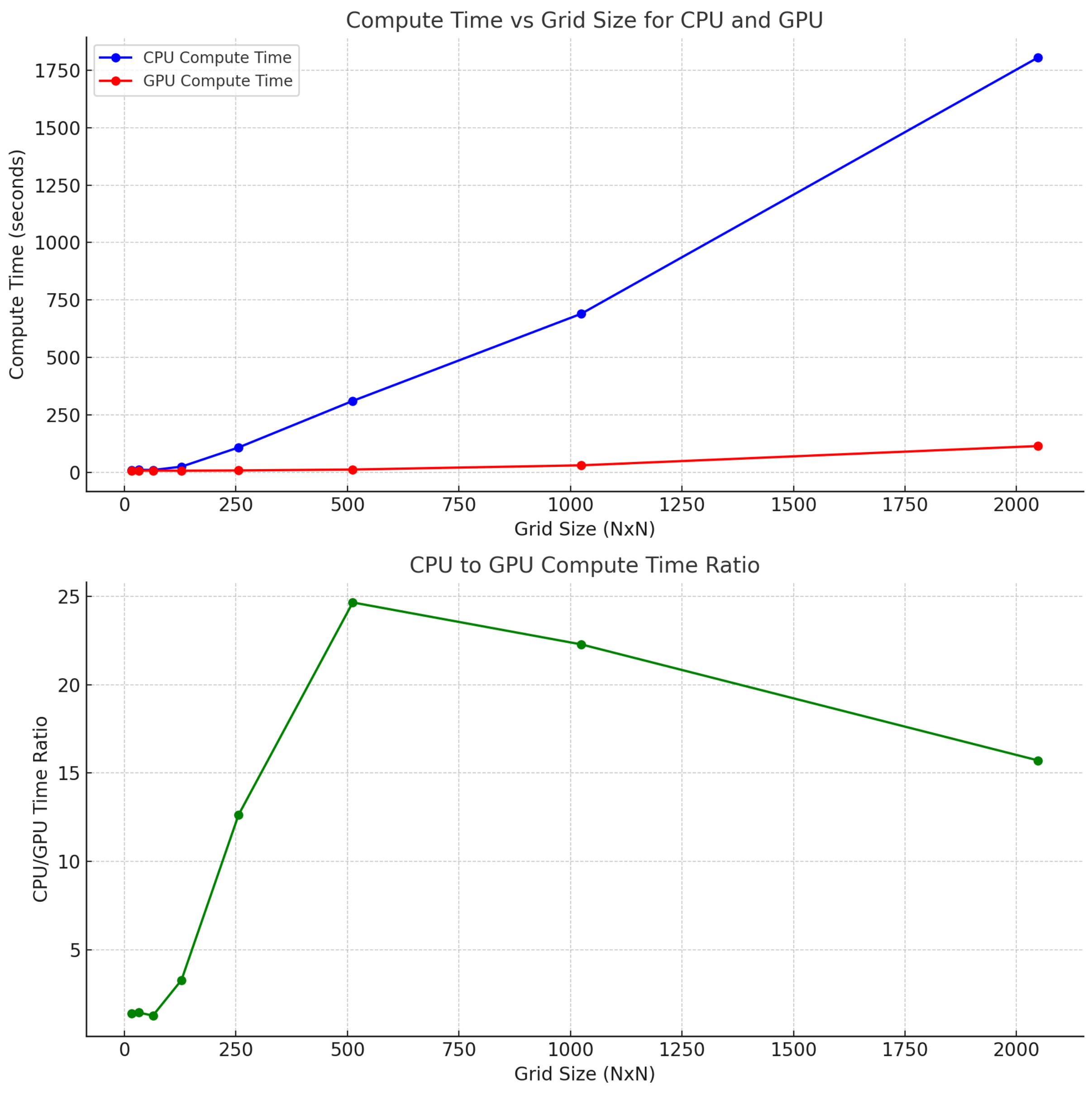

Figure 7: Plot of the Performance difference between the CPU and GPU

From the figure, we can see how much the GPU outperforms the CPU in this simulation, by a factor of up to 25 in the best-case scenario. However, we observe a decrease in the performance ratio as the grid size increases because the GPU utilization eventually maxes out its available VRAM, leading to a bottleneck in processing power. In the `Oceananigans.jl` paper, this phenomenon is also noted, where large-scale simulations approach the GPU's memory limits, impacting overall performance.

5 Conclusions

This report explored fluid dynamics simulations using `Oceananigans.jl`, focusing on its GPU-accelerated capabilities for computationally intensive models like the Boussinesq equations. The study highlighted the model's effectiveness in simulating phytoplankton convection and fluid simulations. By capturing the conditions leading to spring phytoplankton blooms, the model demonstrated the critical role of buoyancy flux reduction in triggering bloom initiation, aligning with the critical turbulence hypothesis by Taylor and Ferrari (2011) [TF11].

The use of GPU computing significantly enhanced performance, especially for large-scale simulations, outperforming CPU-based methods. The results show that `Oceananigans.jl` is a powerful tool for high-resolution oceanographic simulations, providing both accuracy and efficiency in modeling complex processes like phytoplankton dynamics.

References

[Fan+22] Kezhao Fang et al. “Boussinesq Simulation of Coastal Wave Interaction with Bottom-Mounted Porous Structures”. In: *Journal of Marine Science and Engineering* 10.10 (2022). ISSN: 2077-1312. DOI: 10.3390/jmse10101367. URL: https://www.mdpi.com/2077-1312/10/10/1367.

[Kar12] Sajal K. Kar. “An Explicit Time-Difference Scheme with an Adams–Bashforth Predictor and a Trapezoidal Corrector”. In: *Monthly Weather Review* 140.1 (2012), pp. 307–322. DOI: 10.1175/MWR-D-10-05066.1. URL: https://journals.ametsoc.org/view/journals/mwre/140/1/mwr-d-10-05066.1.xml.

[LCL10] Katherine Lundquist, Fotini Chow, and Julie Lundquist. “An Immersed Boundary Method for the Weather Research and Forecasting Model”. In: *Monthly Weather Review* 138 (Mar. 2010), pp. 796–817. DOI: 10.1175/2009MWR2990.1.

[Sil+23] Simone Silvestri et al. *Oceananigans.jl: A model that achieves breakthrough resolution, memory and energy efficiency in global ocean simulations*. 2023. arXiv: 2309.06662 [physics.ao-ph]. URL: https://arxiv.org/abs/2309.06662.

[Sil+24] Simone Silvestri et al. “A New WENO-Based Momentum Advection Scheme for Simulations of Ocean Mesoscale Turbulence”. In: *Journal of Advances in Modeling Earth Systems* 16.7 (2024). e2023MS004130 2023MS004130, e2023MS004130. DOI: https://doi.org/10.1029/2023MS004130. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2023MS004130. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023MS004130.

[TF11] John R. Taylor and Raffaele Ferrari. “Shutdown of turbulent convection as a new criterion for the onset of spring phytoplankton blooms”. In: *Limnology and Oceanography* 56.6 (2011), pp. 2293–2307. DOI: https://doi.org/10.4319/lo.2011.56.6.2293. eprint: https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lo.2011.56.6.2293. URL: https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.2011.56.6.2293.

A Code samples

```julia
grid = RectilinearGrid(GPU(), size=(64, 64), extent=(64, 64), halo=(3, 3), topology=(

# Boundary conditions
buoyancy_flux(x, t, params) = params.initial_buoyancy_flux * exp(-t^4 / (24 * params.
buoyancy_flux_parameters = (initial_buoyancy_flux = 1e-8, shut_off_time = 2hours)
buoyancy_flux_bc = FluxBoundaryCondition(buoyancy_flux, parameters = buoyancy_flux_pa

# Plot buoyancy flux over time
times = range(0, 12hours, length=100)
fig = Figure(size = (800, 300))
ax = Axis(fig[1, 1]; xlabel = "Time (hours)", ylabel = "Surface buoyancy flux (m² s³)
flux_time_series = [buoyancy_flux(0, t, buoyancy_flux_parameters) for t in times]
lines!(ax, times ./ hour, flux_time_series)
fig

# Define boundary conditions
N² = 1e-4 # s²
buoyancy_gradient_bc = GradientBoundaryCondition(N²)
buoyancy_bcs = FieldBoundaryConditions(top = buoyancy_flux_bc, bottom = buoyancy_grad

# Phytoplankton dynamics
growing_and_grazing(x, z, t, P, params) = (params. * exp(z / params.) - params.m) * P
plankton_dynamics_parameters = ( = 1/day,  = 5, m = 0.1/day)
plankton_dynamics = Forcing(growing_and_grazing, field_dependencies = :P, parameters

# Set up the model
model = NonhydrostaticModel(
    grid = grid,
    advection = UpwindBiasedFifthOrder(),
    timestepper = :RungeKutta3,
    closure = ScalarDiffusivity(=1e-4, =1e-4),
    coriolis = FPlane(f=1e-4),
    tracers = (:b, :P),
    buoyancy = BuoyancyTracer(),
    forcing = (; P = plankton_dynamics),
    boundary_conditions = (; b = buoyancy_bcs)
)

# Initial conditions
mixed_layer_depth = 32 # m
stratification(z) = z < -mixed_layer_depth ? N² * z : -N² * mixed_layer_depth
noise(z) = 1e-4 * N² * grid.Lz * randn() * exp(z / 4)
initial_buoyancy(x, z) = stratification(z) + noise(z)
set!(model, b = initial_buoyancy, P = 1)
```

```julia
# Set up the simulation
simulation = Simulation(model, t = 2minutes, stop_time = 24hours)

# Adaptive time stepping
conjure_time_step_wizard!(simulation, cfl = 1.0, max_t = 2minutes)

# Progress callback
function progress(sim)
    @printf("Iteration: %d, time: %s, t: %s\n", iteration(sim), prettytime(sim), pret
end
add_callback!(simulation, progress, IterationInterval(100))

# Output writer
outputs = (w = model.velocities.w, P = model.tracers.P, avg_P = Average(model.tracers
simulation.output_writers[:simple_output] = JLD2OutputWriter(model, outputs, schedule

# Run the simulation
run!(simulation)
```