Region	Horizontal	Vertical	Square
Run-time (s)	82.2	84.34	84.31

Table 1: Run-times for MPI solution of heat equation.

Wave Equation using MPI

We consider the numerical approximation of the wave equation

$$\partial_{tt}u = \partial_{xx}u + \partial_{yy}u \qquad x, y \in (-1, 1), t \in (0, 1)$$

subject to

$$\begin{aligned} u(x, y, t = 0) &= e^{-40((x-0.4)^2+y^2)}; \\ \partial_t u(x, y, t = 0) &= 0; \\ u(x = \pm 1, y, t) &= u(x, y = \pm 1, t) = 0. \end{aligned}$$

A numerical scheme to approximate the solution by $U^n_{i,j} \approx u(-1 + i\Delta x, -1 + i\Delta y, n\Delta t)$ is given by

$$\frac{U^{n+1}_{i,j} - 2U^n_{i,j} + U^{n-1}_{i,j}}{\Delta t^2} = \frac{U^n_{i+1,j} + U^n_{i-1,j} + U^n_{i,j+1} + U^n_{i,j-1} - 4U^n_{i,j}}{\Delta x^2},$$

where $1 \le i, j \le M - 1$, $\Delta x = \Delta y = 2/M$, $0 \le n \le N$ and $\Delta t = 1/N$.

We consider halo-swapping by dividing the domain into rectangular regions. As with the heat equation, we use halo-swapping to update points on the boundary of two regions. For example, division of the plane using horizontal regions is shown in Figure 1. Here, we see an upper (blue) process and lower (red) process, where halo-swapping requires both processes contain the yellow points. The upper row of yellow points are updated via the blue process and sent to the red process at the end of the iteration, whereas the lower row of yellow points are computed via the red process and sent to the blue process. As with the wave equation, given M, the number of rows that should be supplied to each process to spread the cost equally per level is

$$J = \frac{M - 2}{size} + 2,$$

where $size$ is the number of available processes. Additionally, we consider vertical halo-swapping, which transposes the horizontal case. We also consider division of the x, y-domain into equally sized squares. For this,w e require that there $size$ is a square number and take

$$J = \frac{M - 2}{\sqrt{size}} + 2,$$

points for each processor in the x and y direction, halo swapping with all neighboring regions. The solution of the wave equation for $t = 0, 1/3, 2/3, 1$, approximated using horizontal halo swapping, is shown in Figure 2.

In order to investigate the effect of halo-swapping versus the number of processors, we consider an analysis of strong and weak scaling of the problem. For strong scaling, we fix the overall computational cost of the problem (taking $M = 2306$, $N = 0.2/(M - 2)$) and vary the number of process $size$. The runtime of each method for these parameters using 16 processors is shown in Table 1. We consider the parallel speed-up and efficiency for strong scaling, given by

$$\text{speed up}(size) = \frac{\text{runtime for serial code}}{\text{runtime using } size \text{ processes}}, \quad \text{parallel efficiency}(size) = \frac{\text{speed up}(size)}{size}.$$

Ideally, we want the speed up to increase proportionally to $size$ and the efficiency thus to be constant, however this is not the case in practice. The speed-up and efficiency versus number of processors for each halo-splitting method is shown in Figure 3. Note that we observe a (optimal) linear increase in speed-up

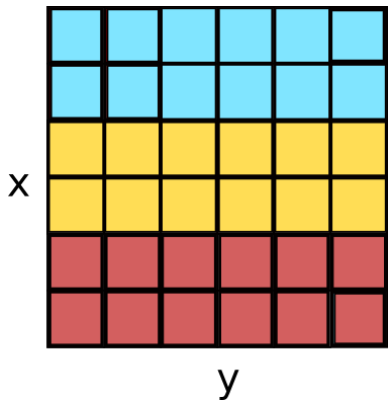

Figure 1: Illustration of horizontal halo-swapping between blue and red region with shared points shown in yellow.

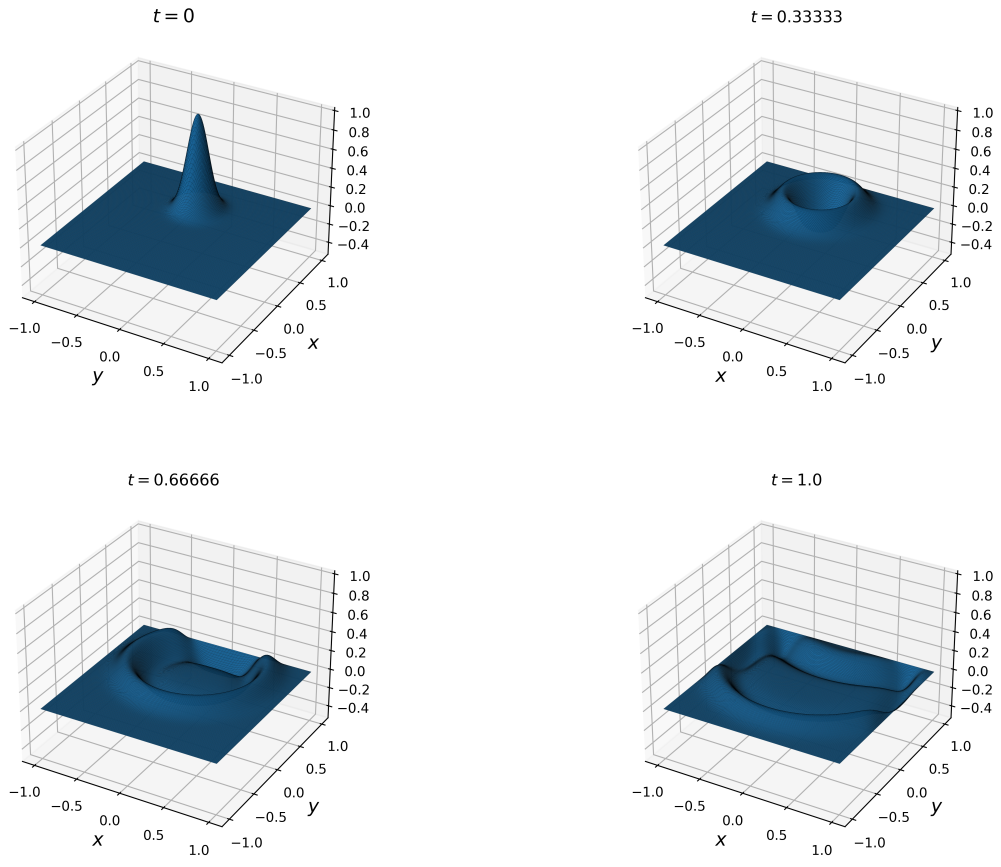

Figure 2: Approximation of u at 4 time points using horizontal halo swapping.

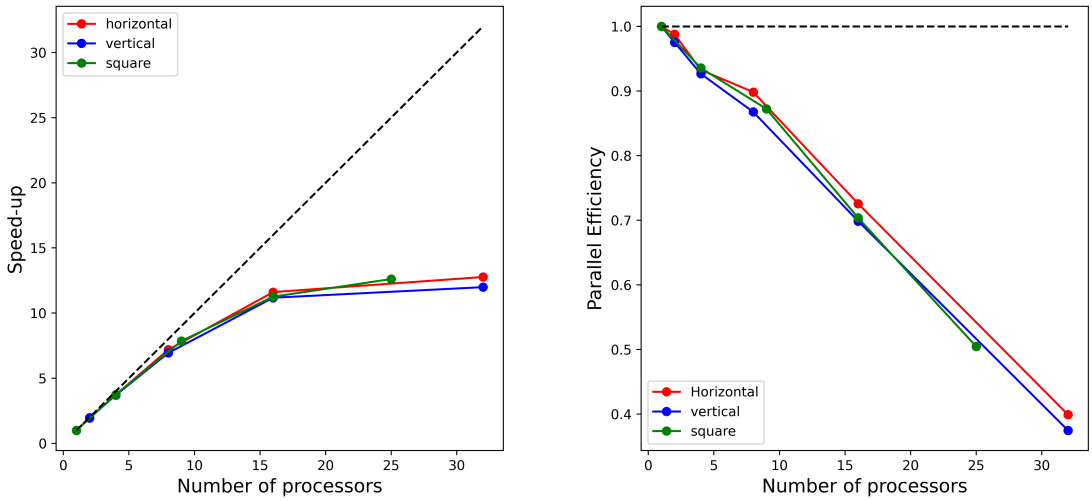

Figure 3: Speed-up (left) and parallel efficiency (right) for the horizontal halo-swapping with strong scaling.

only for up to around 8 processors. The benefit of using more processors then slowly decreases, so that using 32 processors only gives a marginally larger speed-up than 16 processors. This is mirrored by a noticeable decrease in efficiency for $size \geq 16$. Note the slight decrease in speed-up and efficiency for vertical splitting despite the same problem size, especially for larger $size$. This is due to the fact that C++uses row-major storing of arrays. Therefore, the process of sending rows of the solution matrix (horizontal splitting) is faster than sending columns (vertical splitting) since elements of the same row are stored in contiguous memory locations. We see that square splitting has similar speed-up to the Horizontal splitting, but has a lower efficiency. Given the difficulty implementing square splitting, induced by having several boundaries for each region, it seems sensible to favor horizontal splitting. However, as M increases, the storage requirements of each process increase due to the cost of storing a single row of M elements. This can be combated in square splitting by simply adding more processes to hold the extra elements.

With weak scaling, we scale the size of the problem proportional to the number of processes. To do so for horizontal halo-swapping we continue using a grid with $M = 2306$ steps in the y-direction, and $\Delta t = 2/M$. However, we now use

$$M_x = 72 \cdot size + 2$$

steps in the x-direction. For this problem, the optimal performance is for the runtime to remain constant. The runtime versus $size$ for weak scaling and horizontal splitting is shown in Figure 4. Observe that we see roughly constant runtime using $1 \leq size \leq 16$, but see a dramatic increase when using 32 processors. This mirrors the fact that there was almost identical speed up using 16 and 32 processors.

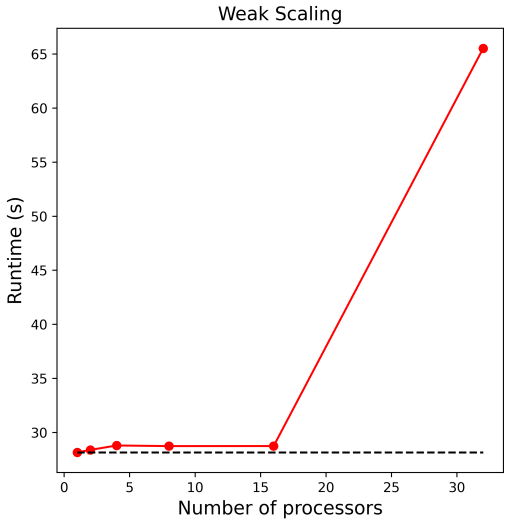

Figure 4: Run-time using weak scaling vs *size* for the horizontal halo-swapping.