Enhancing Computational Efficiency through Parallelization: A Case Study with the Advection Solver

Muhammed Furkan Çanga 2168938
Bertan Özbay 2378545

5.12.2023

1 Introduction

In the realm of computational physics, the accurate simulation of fluid dynamics often relies on sophisticated numerical solvers. The advection equation, describing the transport of a quantity by fluid flow, is a fundamental component of such simulations. This report explores the impact of parallelization on the performance of an advection solver implemented in *C*, utilizing OpenMP directives. The focus is on the impact of parallelizing the computation method, specifically within the RhsQ function, on solver efficiency.

2 Parallelizing the Advection Solver

The advection solver is implemented in C, employing a finite difference method for the numerical solution of the advection equation. The solver incorporates mesh creation based on user-defined parameters and time integration using the Runge-Kutta scheme. Mesh creation establishes the spatial domain and connectivity, while the time integration scheme ensures accurate and stable solutions over successive time steps. The key function, RhsQ, calculates the right-hand side values crucial for advancing the solution in time. This function becomes the focal point for parallelization efforts.

Within the RhsQ function, OpenMP directives are strategically placed to parallelize the computation. The number of threads is set using omp_set_num_threads, and the #pragma omp parallel for directive instructs the compiler to distribute the iterations of the outer loop among the available threads. Private and shared clauses define the scope of variables, ensuring data consistency and avoiding race conditions.

Norm computation is a critical aspect of assessing the accuracy and convergence of the solver. Three different reduction approaches—atomic, critical, and reduction—are employed for calculating the infinity norm. The infinityNormAtomic, infinityNormCritical, and infinityNormReduction functions provide timings for each approach, allowing a comparative analysis of their computational efficiency.

3 Results

Results of 201 × 201 Mesh Grid with 2 Threads Usage			
STEP	METHOD	RESULT	TIME(s)
1	Atomic	0.751354	0.000383
	Critical	0.751354	0.000175
	Reduction	0.751354	0.000074
2	Atomic	2.294398	0.000119
	Critical	2.294398	0.000079
	Reduction	2.294398	0.000078
3	Atomic	7.310563	0.000112
	Critical	7.310563	0.000082
	Reduction	7.310563	0.000081
4	Atomic	17.199645	0.000111
	Critical	17.199645	0.000078
	Reduction	17.199645	0.000076
5	Atomic	29.985218	0.000154
	Critical	29.985218	0.000077
	Reduction	29.985218	0.000076
6	Atomic	43.465573	0.000114
	Critical	43.465573	0.000090
	Reduction	43.465573	0.000090
7	Atomic	56.673202	0.000199
	Critical	56.673202	0.000155
	Reduction	56.673202	0.000167
8	Atomic	69.273548	0.000146
	Critical	69.273548	0.000096
	Reduction	69.273548	0.000093
9	Atomic	81.151798	0.000121
	Critical	81.151798	0.000094
	Reduction	81.151798	0.000094
10	Atomic	92.289794	0.000114
	Critical	92.289794	0.000090
	Reduction	92.289794	0.000090

Table 1 Results of 201×201 Mesh Grid with 2 Threads Usage

Results of 401 × 401 Mesh Grid with 2 Threads Usage			
STEP	METHOD	RESULT	TIME
1	Atomic	0.751354	0.000791
	Critical	0.751354	0.000631
	Reduction	0.751354	0.000290
2	Atomic	2.305377	0.000576
	Critical	2.305377	0.000371
	Reduction	2.305377	0.000354
3	Atomic	7.891932	0.000399
	Critical	7.891932	0.000414
	Reduction	7.891932	0.000346
4	Atomic	22.280105	0.000497
	Critical	22.280105	0.000419
	Reduction	22.280105	0.000407
5	Atomic	45.273270	0.000419
	Critical	45.273270	0.000372
	Reduction	45.273270	0.000354
6	Atomic	71.844684	0.000474
	Critical	71.844684	0.000385
	Reduction	71.844684	0.000618
7	Atomic	98.762102	0.000414
	Critical	98.762102	0.000357
	Reduction	98.762102	0.000345
8	Atomic	124.837205	0.000434
	Critical	124.837205	0.000365
	Reduction	124.837205	0.000355
9	Atomic	149.665253	0.000441
	Critical	149.665253	0.000404
	Reduction	149.665253	0.000395
10	Atomic	173.085232	0.000427
	Critical	173.085232	0.000371
	Reduction	173.085232	0.000356

Table 2 Results of 401 × 401 Mesh Grid with 2 Threads Usage

201x201 MESH - 4 Thread			
STEP	METHOD	RESULT	TIME
1	Atomic	0.751354	0.005051
	Critical	0.751354	0.001562
	Reduction	0.751354	0.000289
2	Atomic	2.294398	0.000069
	Critical	2.294398	0.000044
	Reduction	2.294398	0.000042
3	Atomic	7.310563	0.000103
	Critical	7.310563	0.000089
	Reduction	7.310563	0.000131
4	Atomic	17.199645	0.000084
	Critical	17.199645	0.000055
	Reduction	17.199645	0.000076
5	Atomic	29.985218	0.000110
	Critical	29.985218	0.000085
	Reduction	29.985218	0.000096
6	Atomic	43.465573	0.000081
	Critical	43.465573	0.000052
	Reduction	43.465573	0.000051
7	Atomic	56.673202	0.000081
	Critical	56.673202	0.000053
	Reduction	56.673202	0.000051
8	Atomic	69.273548	0.000080
	Critical	69.273548	0.000052
	Reduction	69.273548	0.000051
9	Atomic	81.151798	0.000076
	Critical	81.151798	0.000050
	Reduction	81.151798	0.000049
10	Atomic	92.289794	0.000111
	Critical	92.289794	0.000082
	Reduction	92.289794	0.000081

Table 3 Results of 201×201 Mesh Grid with 4 Threads Usage

4 Conclusion

Parallel Timings with Various Mesh Resolutions:

The recorded timings for the parallelized section of the solver tabulated on Results section. Table-1 and Table-2 demonstrate consistent behavior across various mesh resolutions. As expected, finer resolutions lead to increased computational times due to a higher number of computational nodes. This behavior aligns with the fundamental trade-off in numerical simulations.Higher accuracy often comes at the cost of increased computational demands. It's noteworthy that the solver's scalability across different resolutions allows users to tailor the mesh size to strike a balance between accuracy and computational efficiency as seen in time difference.

Impact of Thread Count:

The impact of varying thread counts on solver efficiency unveils an interesting trend. The solver showcases commendable scalability with an increasing number of threads, indicating effective parallelization. When the thread number is doubled, computation time is significantly decreased as it seen on Table-1 and Table-3. The time didn't scale exactly as the thread number due to overhead and communication times, limited parallelizable regions and thread management overhead.

However, the observed diminishing returns beyond a certain thread count emphasize the importance of considering the underlying hardware architecture. While the solver benefits from parallel processing, optimal performance is achieved by aligning the thread count with the available computational resources. This finding is crucial for practitioners aiming to harness the full potential of parallelization without encountering diminishing returns.

Norm Computation Timings:

The timings for computing norms using different reduction approaches provide valuable insights into the efficiency of each method as it seen in Table-1, Table-2, Table-3. The reduction approach consistently outperforms both atomic and critical methods. This result is particularly significant as it underscores the importance of selecting an appropriate reduction strategy for norm calculations. The reduction method's superior performance is attributed to its efficient parallel reduction capabilities, showcasing the relevance of tailoring the parallelization strategy to the specific computational task. This insight is vital for people relying on accurate norm calculations in their simulations.