

Parallelizing C++ boids simulation with OpenMP

Parallel Computing First Assignment

Giacomo Orsucci

February, 2026



UNIVERSITÀ
DEGLI STUDI
FIRENZE
Da un secolo, oltre.



Project objective

- Implement a first sequential version of Boids Flocking Simulation.
- Experiment various parallel versions using OpenMP.
- Evaluate the impact of AOS vs SOA, padding, SIMD etc etc.

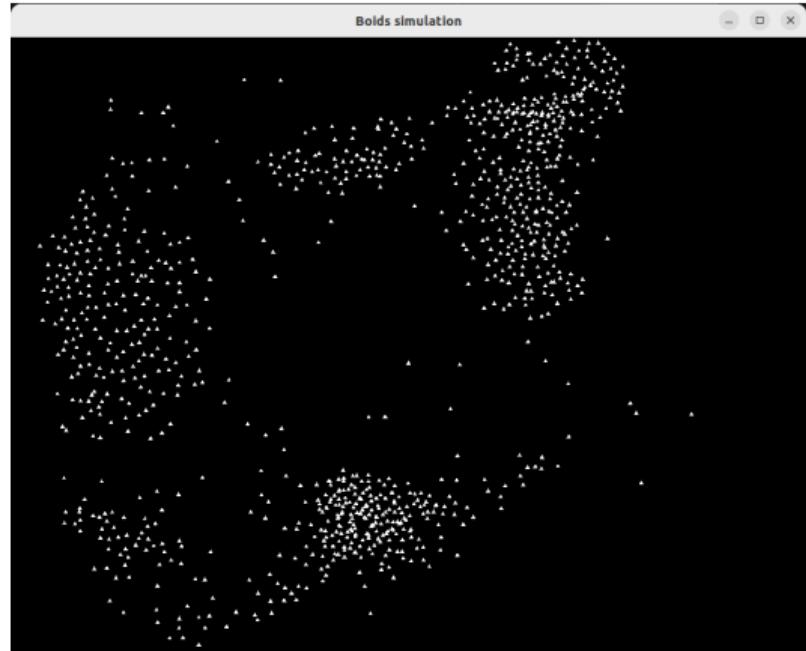


Figure: Graphical boids simulation.



Boids flocking simulation

Three main rules:

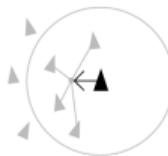
- **Separation:** Boids steer to avoid to be too close to local flockmates.
- **Alignment:** Each boid attempts to match the velocity of other boids inside its visible range.
- **Cohesion:** Each boid steers gently toward the center of mass of other boids within its visible range.



Separation:
Steer to avoid crowding
local flockmates



Alignment:
Steer toward the average
heading of local flockmates



Cohesion:
Steer to move toward the average
position of local flockmates

Figure: The three main rules of boids simulation.



Different implementations and experiments

AOS

```
1  struct Boid {  
2      float x, y;  
3      float vx, vy;  
4  };  
5  std::vector<Boid> boids(N);  
6
```

SOA

```
1  struct Boids {  
2      float *x, *y;  
3      float *vx, *vy;  
4  };  
5  
6
```



Different implementations and experiments

Alignment

```
1 struct alignas(CACHE_SIZE) Boid
2 {
3     float x, y;
4     float vx, vy;
5 }
6
```

Internal padding

```
1 struct alignas(CACHE_SIZE) Boid
2 {
3     float x, y;
4     float vx, vy;
5     char padding[32 - sizeof(float)
6                           * 4];
7 }
8
9
```

In addition to SIMD and Reduction directives.



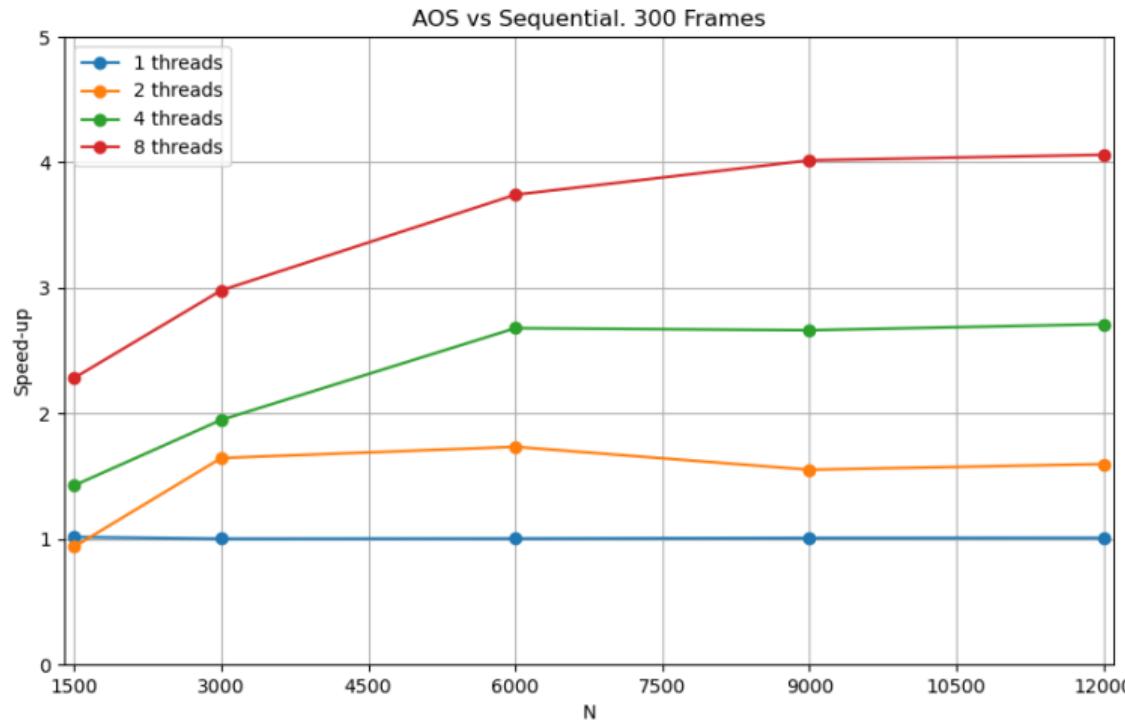
Machine Specifications

- **CPU:** AMD Ryzen 7 3700U, 4 cores, 8 threads, base clock 2.3 GHz, max boost clock 4.0 GHz.
- **Cache:**
 - L1 Cache: 96 KB per core.
 - L2 Cache: 512 KB per core.
 - L3 Cache: 4 MB shared.
- **RAM:** 20 GB DDR4 (1 x 4 GB SODIMM DDR4 + 1 x 16 GB Row of Chips DDR4) at 2400 MHz.
- **OS:** Ubuntu 22.04 LTS.
- **Compiler:** GCC 11.4.0.
- **OpenMP:** version 5.2.



Experimental results

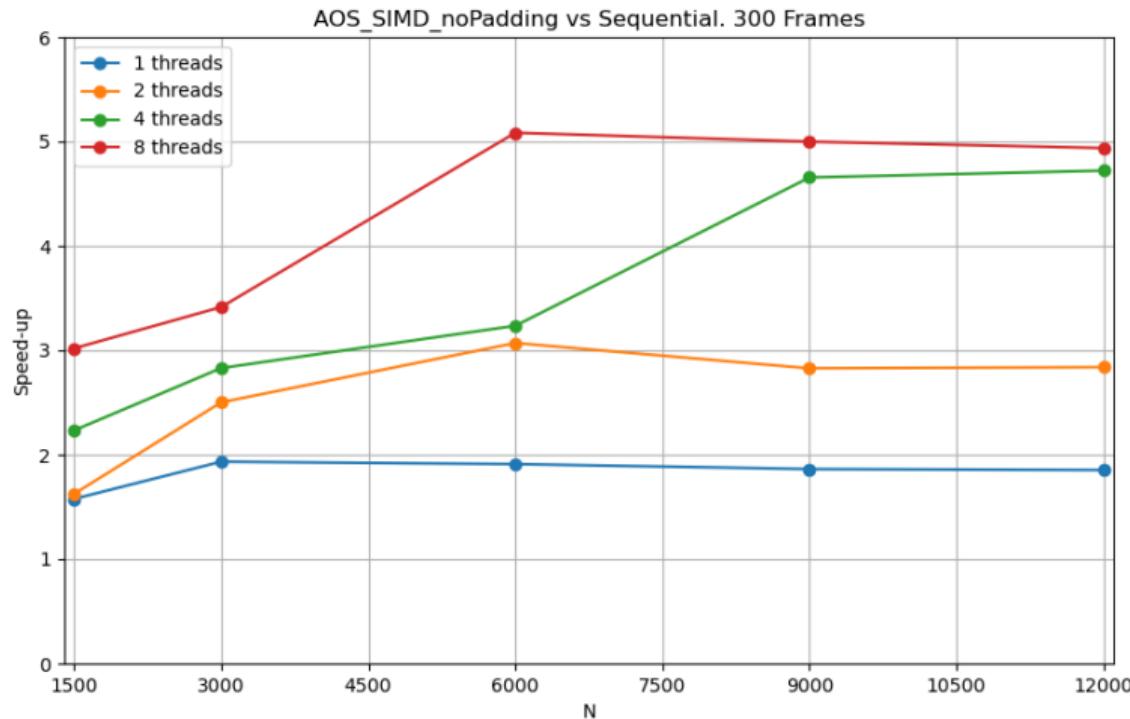
AOS vs Sequential





Experimental results

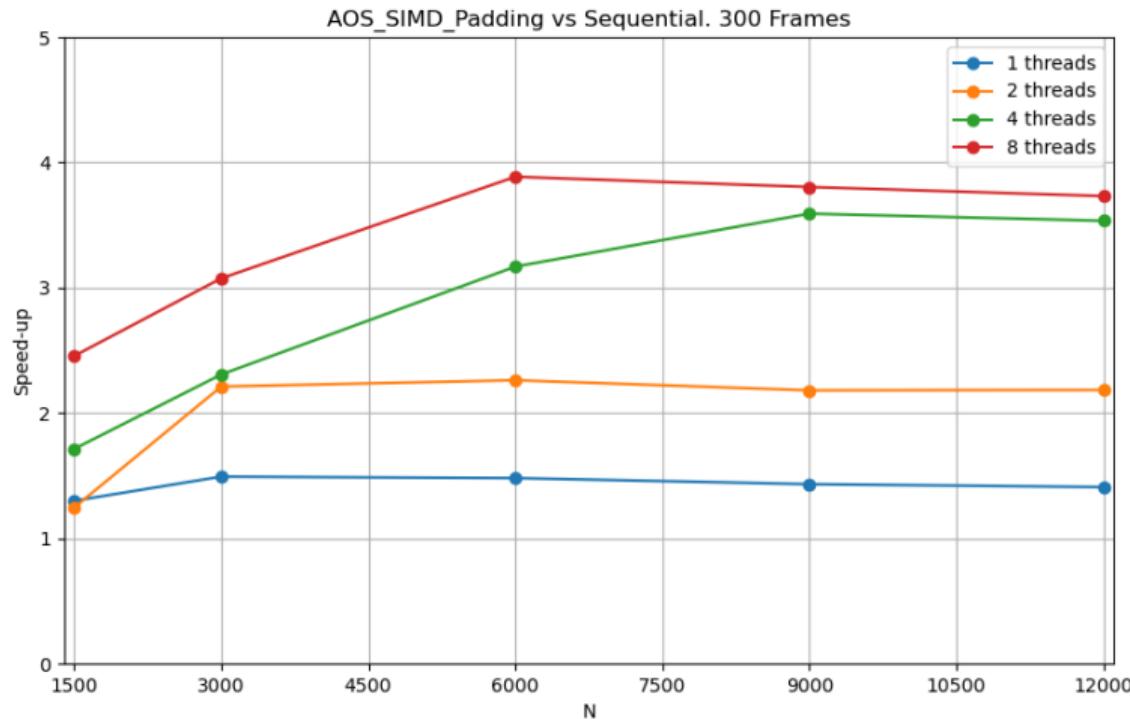
AOS SIMD Alignment vs Sequential





Experimental results

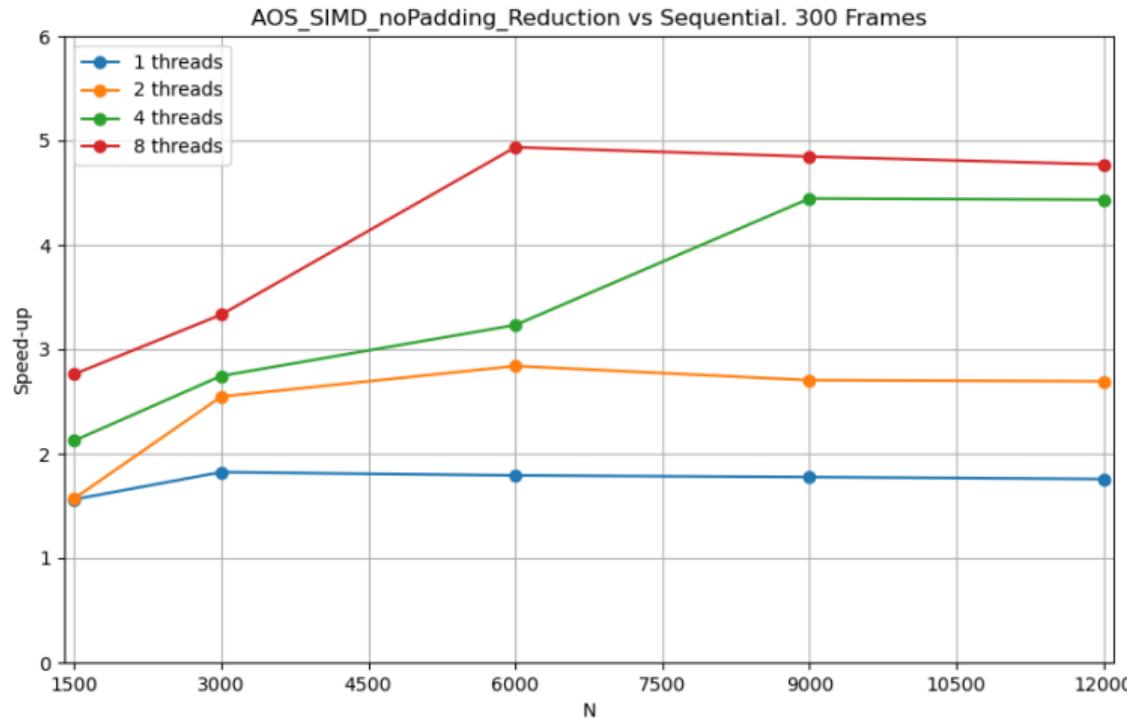
AOS SIMD Padding vs Sequential





Experimental results

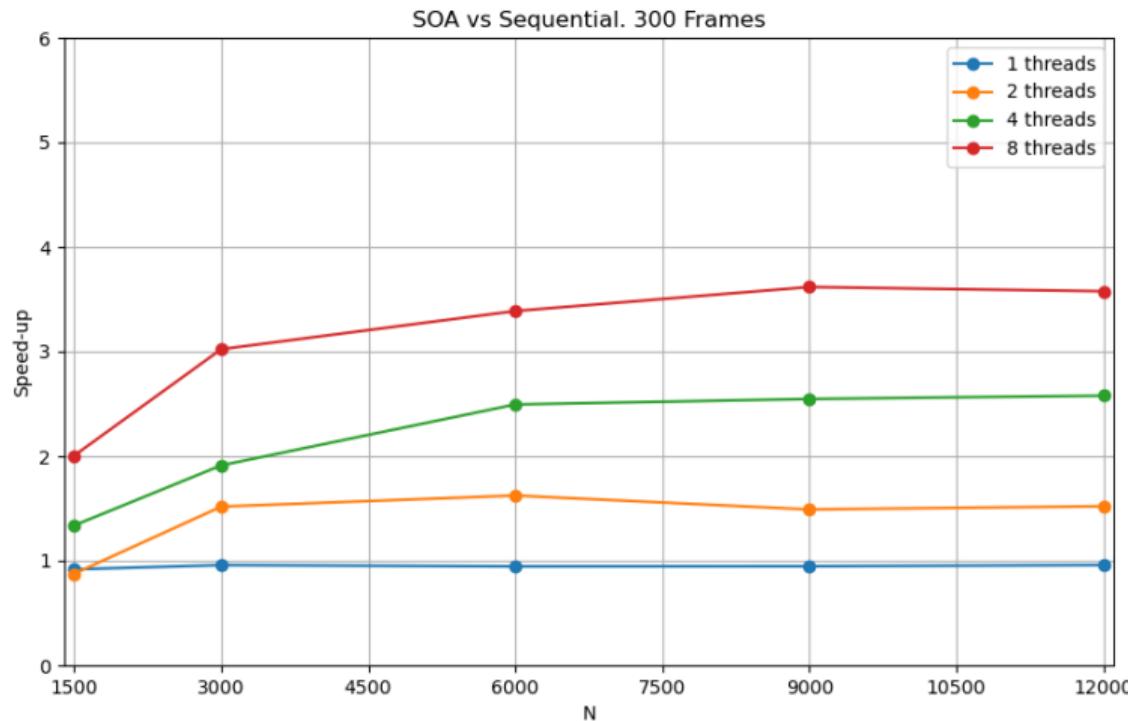
AOS SIMD Alignment Reduction vs Sequential





Experimental results

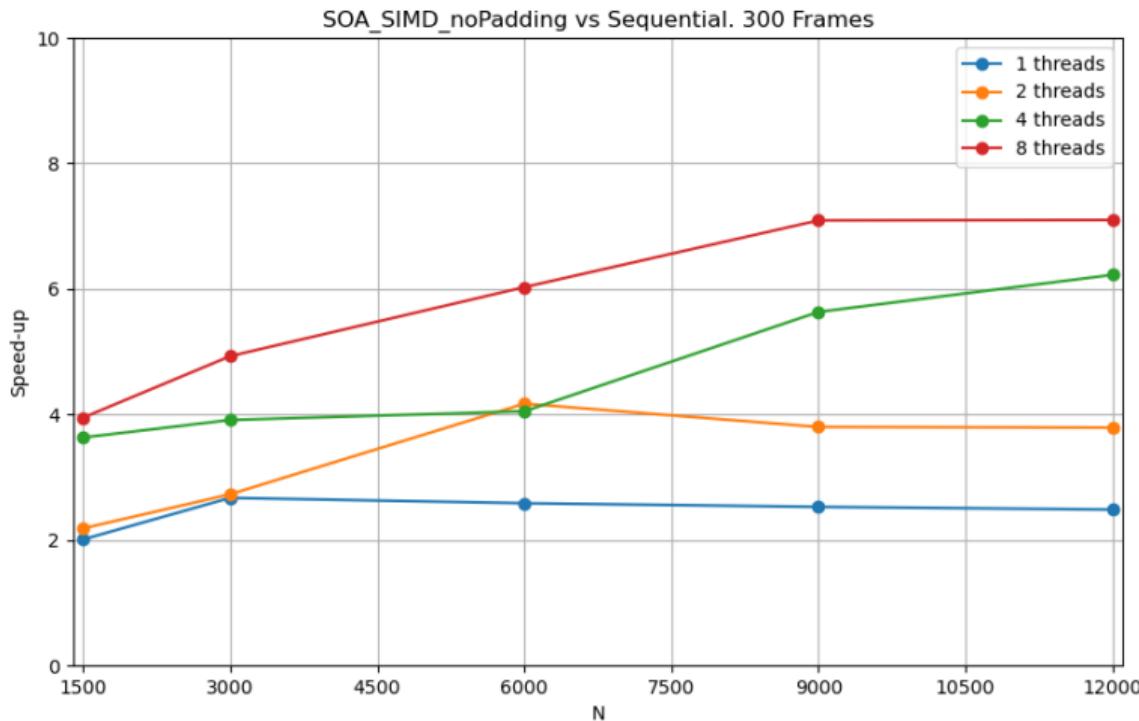
SOA vs Sequential





Experimental results

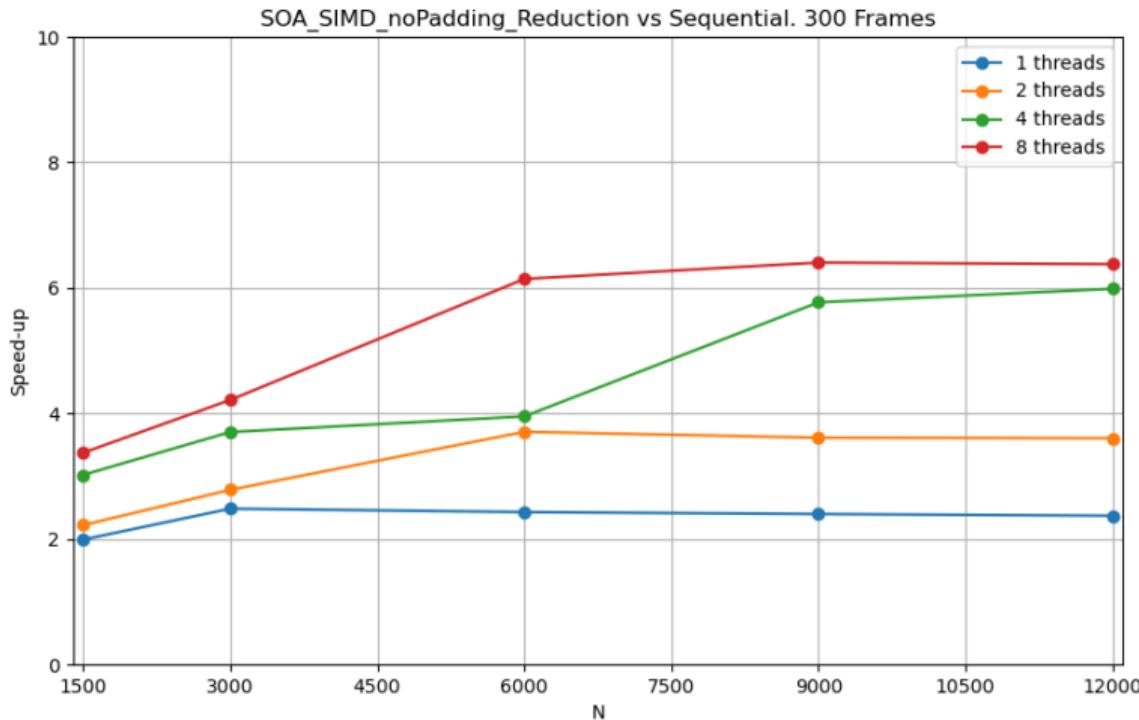
SOA SIMD vs Sequential





Experimental results

SOA SIMD Reduction vs Sequential



Conclusion

- OpenMP parallelization is effective, but its full potential is locked without proper memory layout optimization.
- SoA + SIMD effectively overcomes the memory bandwidth bottlenecks (Memory Wall).
- Hyper-Threading proved beneficial, significantly boosting performance in the Compute-Bound scenario.
- Internal padding doesn't improve performance.