Rust AB: Graphics Update & Distributed Flockers

Speaker:
Pasquale Caramante
Francesco Foglia

Rust-AB

Tool ready-to-use per lo sviluppo di simulazioni basate su agenti, realizzato in Rust

Visualization

Framework di visualizzazione delle simulazioni basato sul game engine Bevy

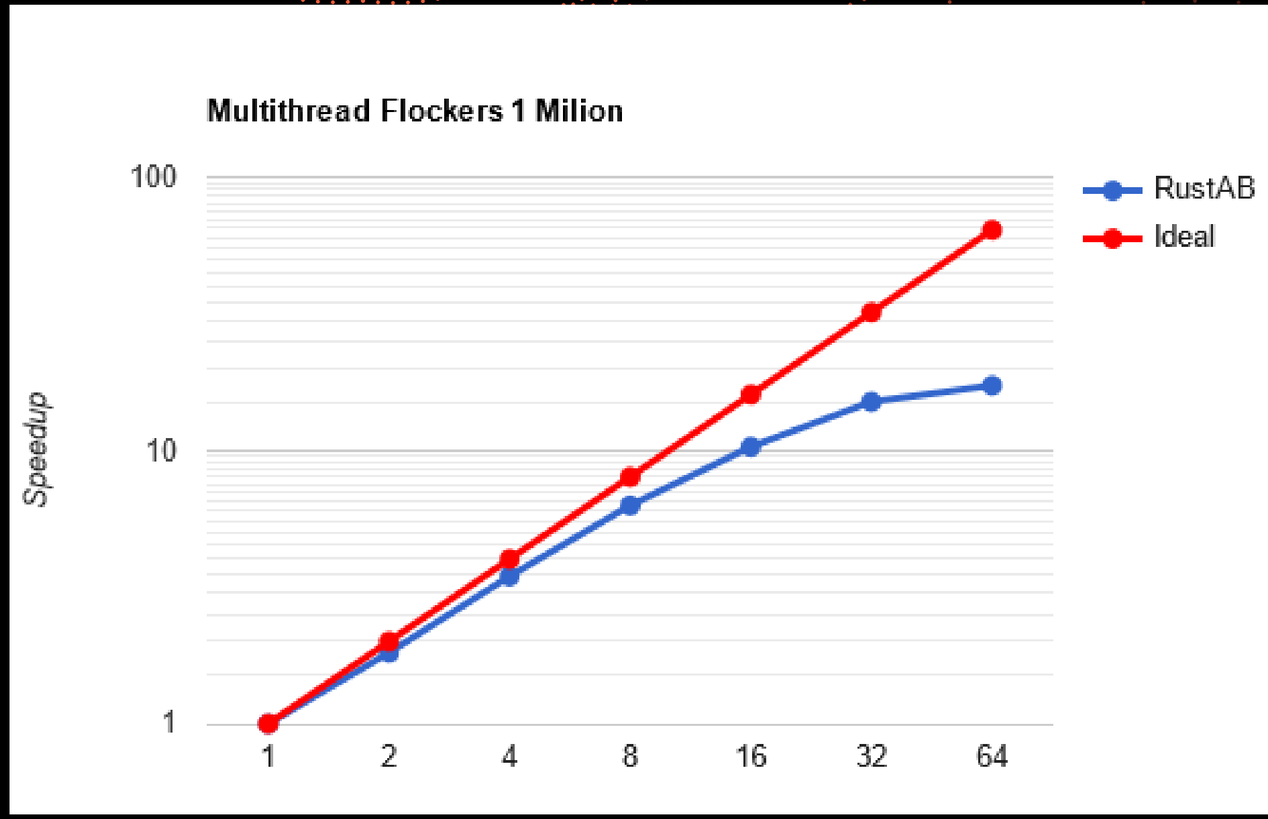
Parallel Execution

Esecuzione parallela delle simulazioni in multithread

Where we left off...

- Shared Memory;
- Speedup non ottimale aumentando il numero di thread;
- Refactoring generale;

N. Thread	Speedup
1	1.00
2	1.82
4	3.46
8	6.29
16	10.30
32	15.05
64	17.27



Flockers (1 milion agents)

...where we're headed

 Testare le prestazioni della libreria in ambiente distribuito

 Parallelizzazione in-model (Flockers) con MPI

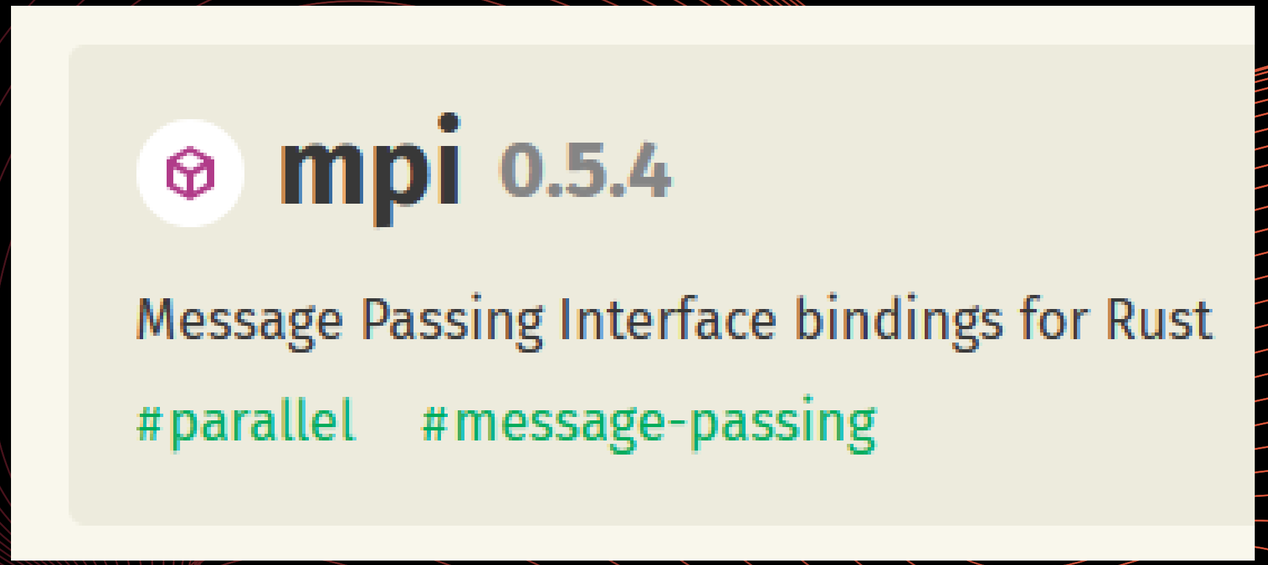
MPI in Rust: rsmpi

Tramite FFI rende disponibili le funzioni MPI in Rust

Supporto per:

- OpenMPI: 3.0.4, 3.1.4, 4.0.1 (?)
- MPI-CH: 3.3, 3.2.1
- MS-MPI: 10.0.0

Dead project? (ultimo commit Marzo 2021) ☹



rsmpi: supported MPI primitives

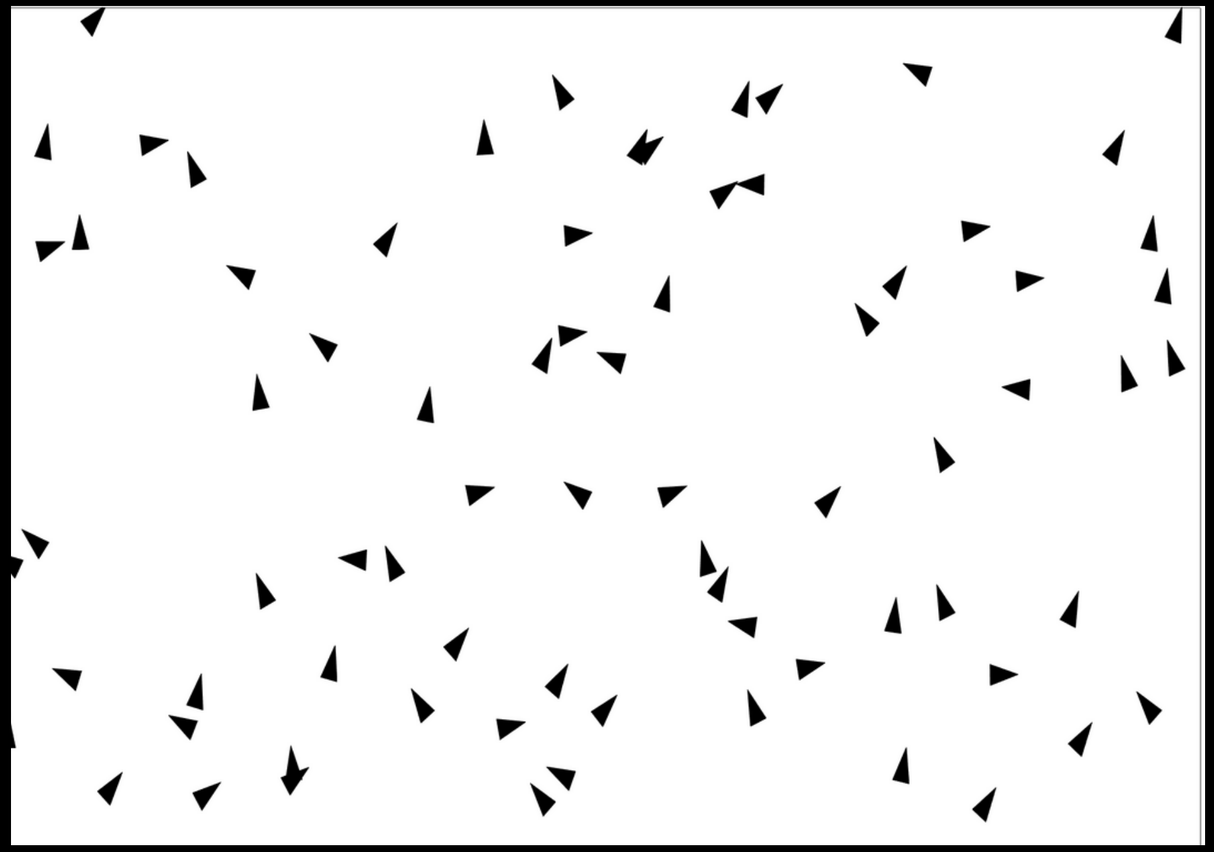
- **Groups, Contexts, Communicators**:
  - Group and (Intra-)Communicator management from section 6 is mostly complete.
  - no Inter-Communicators
  - no process topologies
- **Point to point communication**:
  - standard, buffered, synchronous and ready mode send in blocking and non-blocking variants
  - receive in blocking and non-blocking variants
  - send-receive
  - probe
  - matched probe/receive
- **Collective communication**:
  - barrier
  - broadcast
  - (all) gather
  - scatter
  - all to all
  - varying counts operations
  - reductions/scans
  - blocking and non-blocking variants
- **Datatypes**: Bridging between Rust types and MPI basic types as well as custom MPI datatypes which can act as views into buffers.

Flockers in Rust-AB

- Campo bi-dimensionale (in Double Buffer)

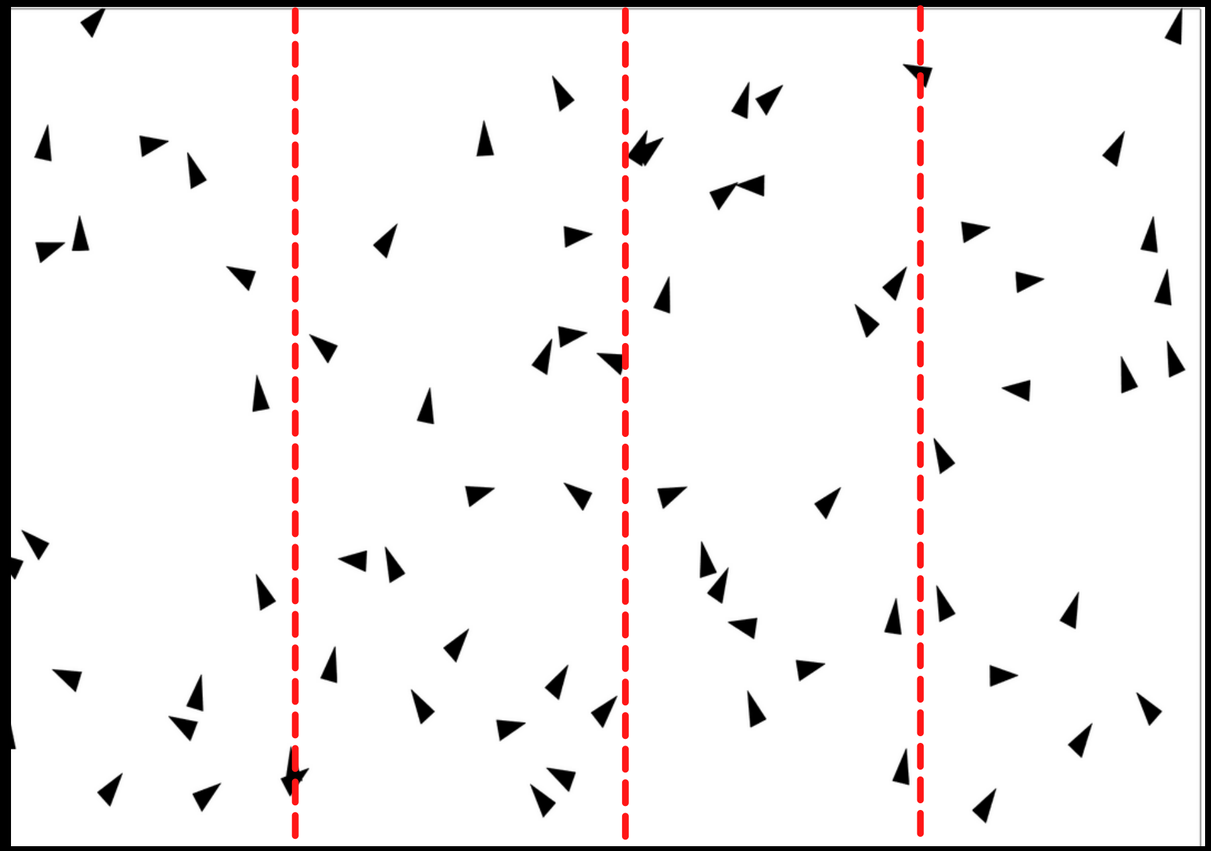
- Ciascun agente ad ogni step di simulazione:
  - legge la posizione dei vicini entro un raggio R;
  - calcola le coordinate della sua nuova posizione (Steering, Alignment, Cohesion);
  - scrive la sua nuova posizione nello stato di simulazione;

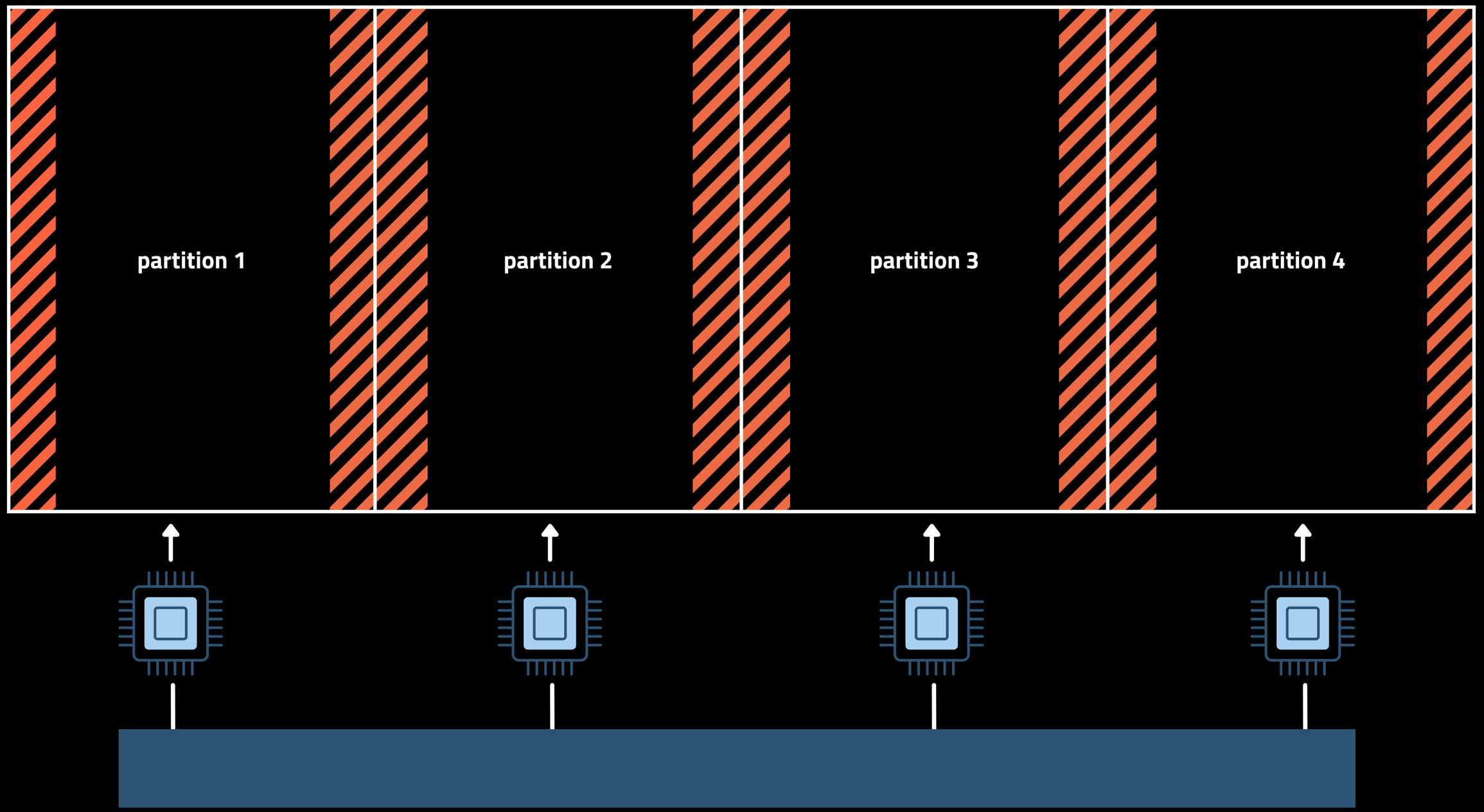
- Il Double Buffer viene aggiornato alla fine dello step di simulazione;



Distributed Flockers

- Partizionamento verticale del campo:
- Start: processo master genera gli agenti e li distribuisce ai processi worker;
- Problemi da gestire:
  - lettura del vicinato;
  - migrazione degli agenti;




partition 1
partition 2
partition 3
partition 4

State (struct e hook after_step)

```rust
pub struct Flocker {
    pub step: u64,
    pub field1: Field2D<Bird>,
    pub dim: (f32, f32),
    pub initial_flockers:u32,
    pub universe: Universe,
    pub partition: (f32,f32),
    pub l_aoi: Mutex<Vec<Bird>>,
    pub r_aoi: Mutex<Vec<Bird>>
}
```

```rust
//processes exchange information about agents around the border (l_aoi and r_aoi)
let s1 = mpi::p2p::send_receive_into(&l_aoi[..],&l_process,&mut r_birds[..],&r_process);
let s2 = mpi::p2p::send_receive_into(&r_aoi[..],&r_process,&mut l_birds[..],&l_process);

let received_r = s1.count(Bird::equivalent_datatype()) as usize;
for mut bird in r_birds.into_iter().take(received_r){
    if bird.migrated {
        bird.migrated = false;
        schedule.schedule_repeating(Box::new(bird), schedule.time + 1.0,0);
    }
    self.field1.set_object_location(bird,bird.pos);
}

let received_l = s2.count(Bird::equivalent_datatype()) as usize;
for mut bird in l_birds.into_iter().take(received_l){
    if bird.migrated{
        bird.migrated = false;
        schedule.schedule_repeating(Box::new(bird), schedule.time + 1.0,0);
    }
    self.field1.set_object_location(bird,bird.pos);
}
```

Logica dell'agente

Left AOI Check

Right AOI Check

```rust
impl Agent for Bird {
    fn step(&mut self, state: &mut dyn State) {
        //lettura del vicinato

        //calcolo della nuova posizione (new_x, new y)

        if new_x <= state.partition.0 + NEIGHBOROOD_RADIUS {
            if new_x < state.partition.0{
                self.migrated = true;
            }
            state.l_aoi.lock().unwrap().push(*self);
        }
        if new_x >= state.partition.1 - NEIGHBOROOD_RADIUS{
            if new_x >= state.partition.1{
                self.migrated = true;
            }
            state.r_aoi.lock().unwrap().push(*self);
        }
        state
            .field1
            .set_object_location(*self, Real2D { x: loc_x, y: loc_y });
    }
```

Benchmark

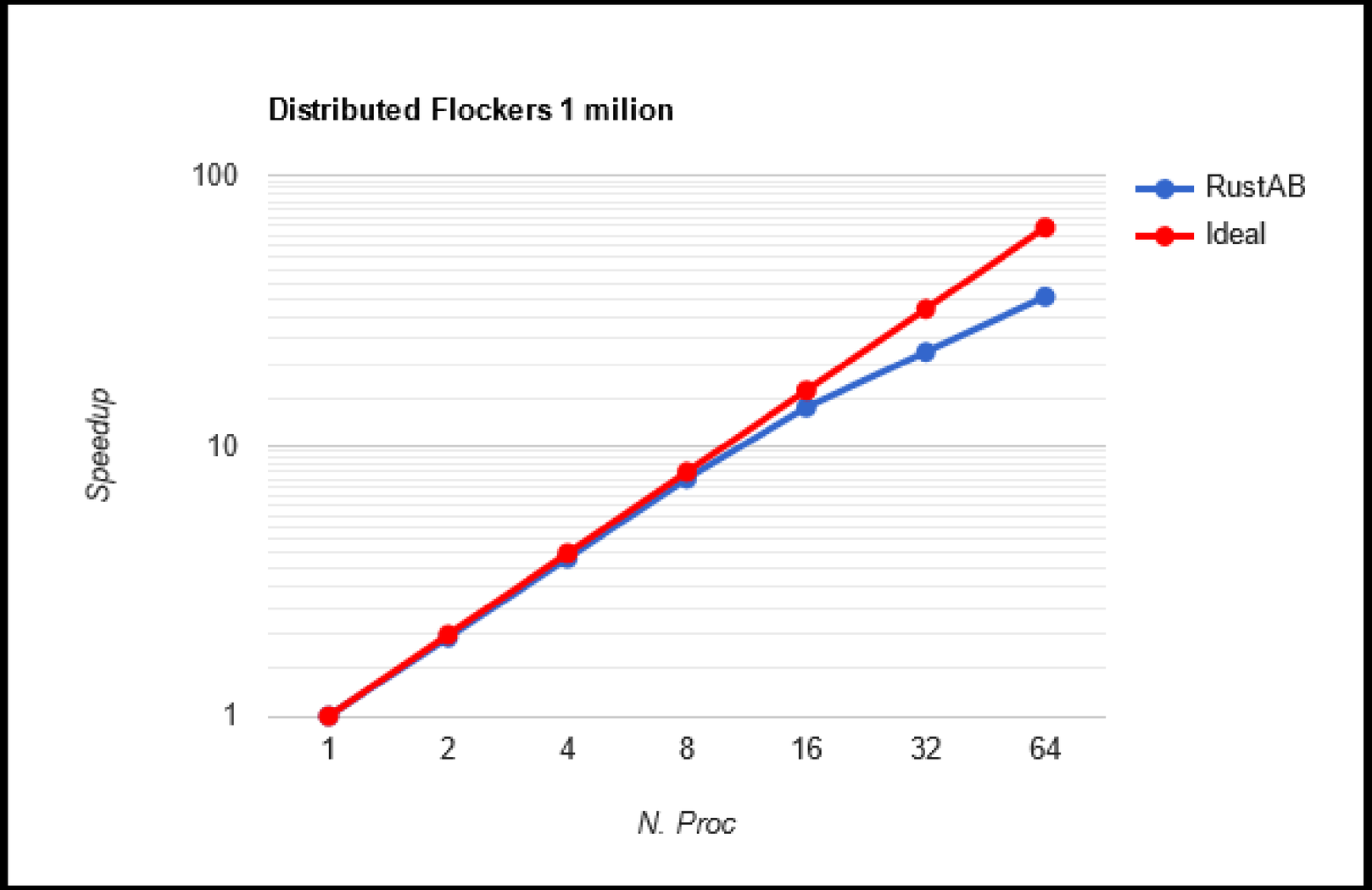
Eseguiti su 16 istanze c5n.2xlarge (4 physical cpu)

Test 1

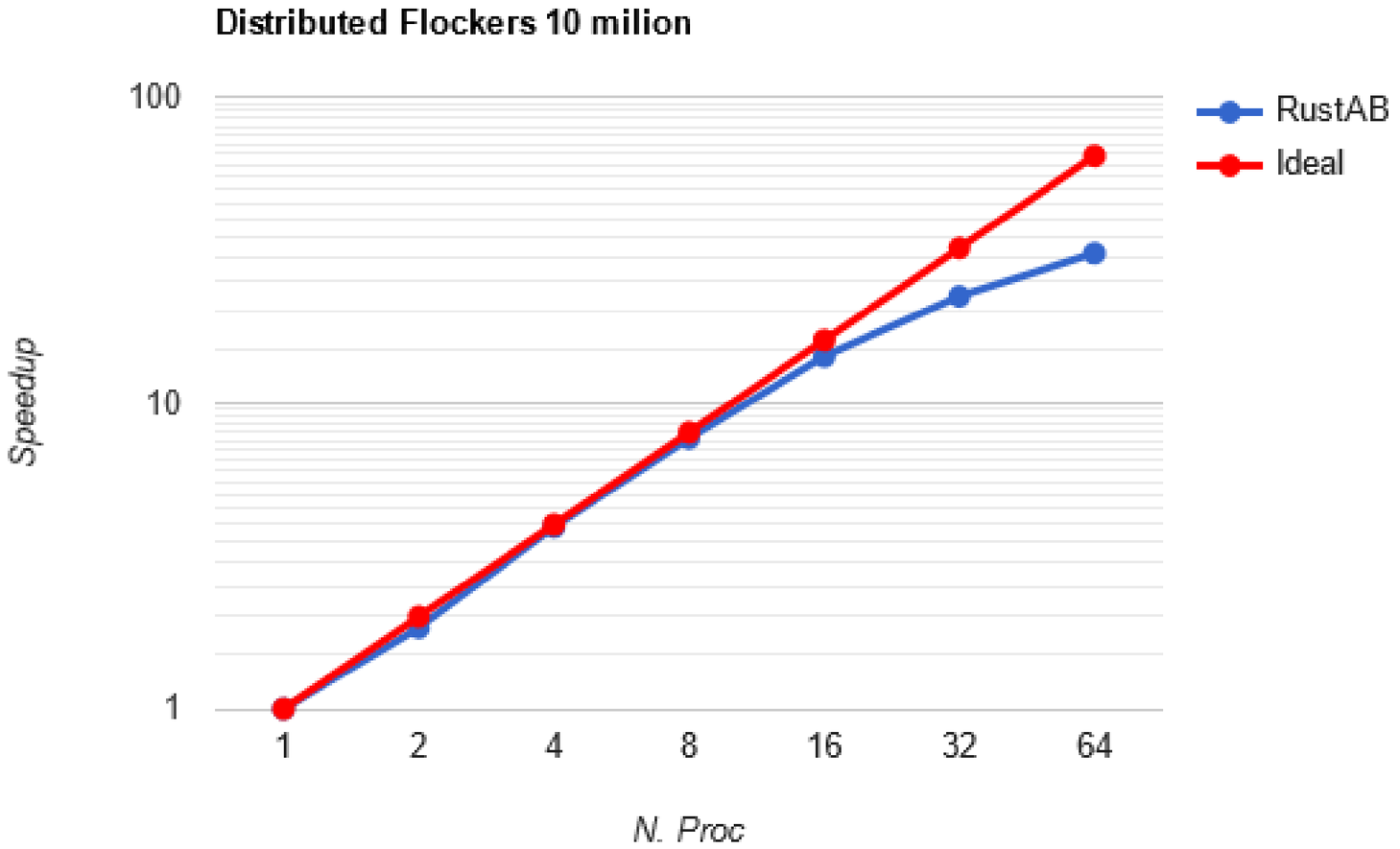
- 1 milione di agenti
- Field 14142.13x14142.13
  - Densità 0.005
- 100 step di simulazione * 10 run

Test 2

- 10 milioni di agenti
- Field 44721.35x44721.35
  - Densità 0.005
- 100 step di simulazione * 10 run

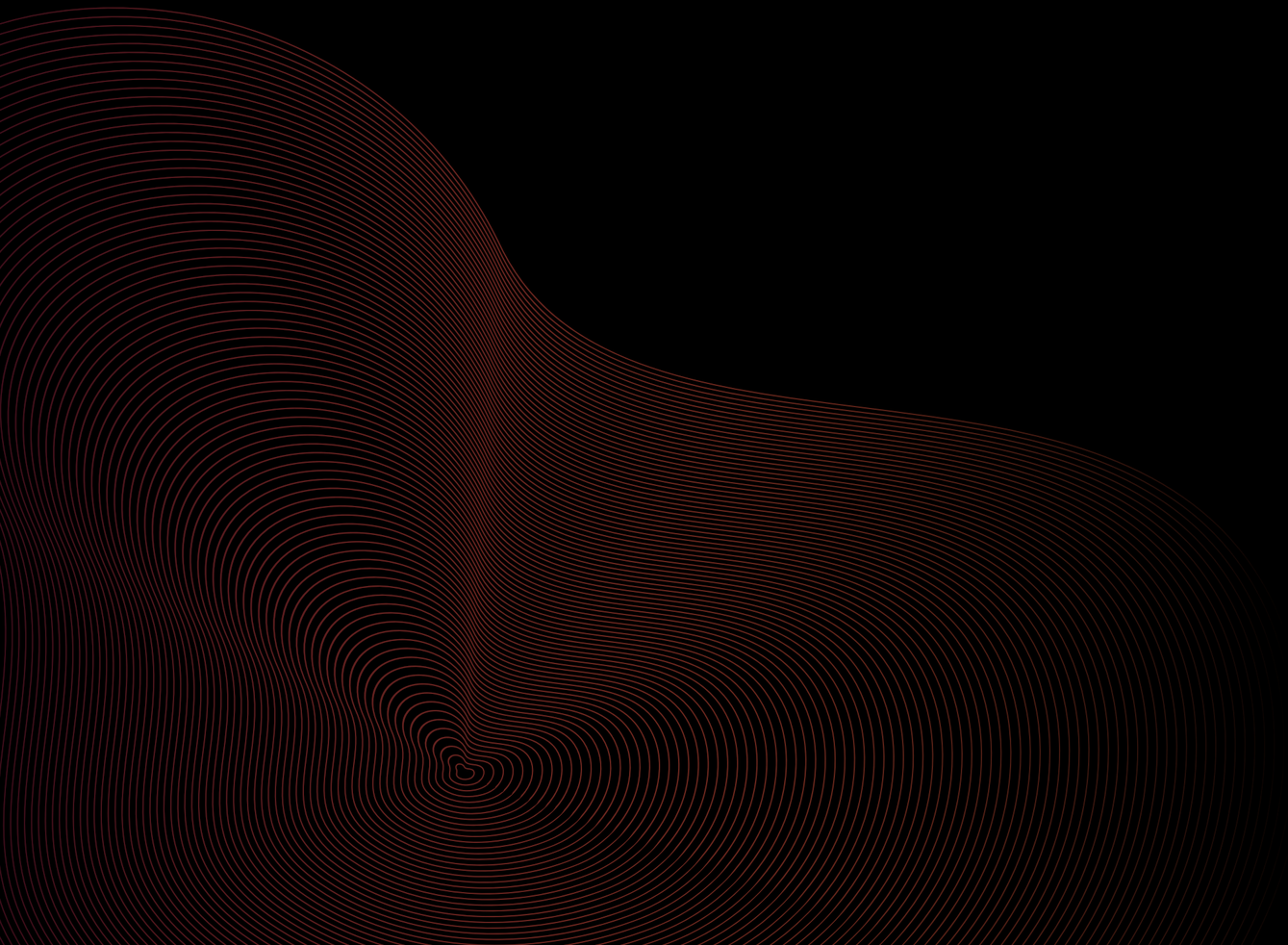


N.Proc	Time (s)	Speedup
1	195.68	1.00
2	100.47	1.95
4	51.03	3.83
8	25.92	7.55
16	14.12	13.86
32	8.81	22.22
64	5.51	35.50



N.Proc	Time (s)	Speedup
1	2410.45	1.00
2	1308.66	1.84
4	613.80	3.93
8	315.23	7.65
16	170.01	14.18
32	108.17	22.28
64	78.29	30.79

Conclusioni



01 Buone performance

02 Implementazione in engine...

03 Implementazione di altri modelli