



#### Kimia Taba and Vincent Hasse

# Predator-Prey Relationship in a Closed System

17.07.2023

University of Göttingen

# Table of contents

- 1 Our Project
- 2 Sequential Approach
- 3 Parallel Approach
- 4 Benchmarking
- 5 Conclusion

## The Idea

- create a simulated environment
- plants, prey, hunters
- apply some machine-learning for prey- and hunter-brains -> Python

# Outline

World: NumPy-Array, int8

- ▶ 0: Free
- 1: Plant
- > 2: Prey
- 3: Hunter
- ... 255 different agents possible
- initialized randomly at the beginning of the simulation, based on config-file
- Agents: Prey and Hunters
  - can move, eat, breed, die







| Our Project | Sequential Approach<br>• | Parallel Approach | Benchmarking | Conclusion |
|-------------|--------------------------|-------------------|--------------|------------|
|             |                          |                   |              |            |

# Pseudo-Code

### Sequential-Simulation in Pseudo-Python

| p | seudo-p | ython  |
|---|---------|--|
| 1 | for     | <pre>step in range(config.SIMUALTION_STEPS):</pre> |
| 2 |         | <pre>for everyAgent in dicts:</pre>                |
| 3 |         | addAge()   |
| 4 |         | moveAgent()  |
| 5 |         | letItDie(energy, age)                              |
| 6 |         | <pre>letItEat(isFoodAtNewPosition)</pre>           |
| 7 |         | letItBreed(energy)                                 |
| 8 |         | spawnNewPlants()                                   |

based on a dict of hunters and a dict of preys: (ID -> Animal)

# General Idea for the Parallel Approach

- Split the world in (even) chunks
- Broadcast world state and Hunter / Prey dicts to every process
- Each process calculates changes in the respective part of the world
- Report changes to rank0
- Rank0 gathers changes, creates new world and dicts and broadcasts for next simulation step



Our Project

# **Problems and Solutions**

Our Project

- Problem: Pickeling & sending of Hunter/ Prey dictionaries is costly
- Solution: don't broadcast the dicts
  - Change dicts from (AnimalID -> Animal) to (position -> Animal)
  - Report back a numpy array with old Positions (as identifier), new Positions
  - -> Broadcast world-state from rank0
  - -> Send changes-array back to rank 0
  - Only communicate NumPy Arrays
- Drawback: rank0 has to do all the updating (world, dicts)

# Parrallel movement calculation

Movement-Calculation in Python; first part

```
#create lists
1
    hunters = []
2
    preys = []
3
    #loop dependent on rank
4
    for x in range((rank-1)*numRows, rank*numRows):
5
        for v in range(config.WORLD_SIZE):
6
            if world[x][y] == config.IDENT_PREY:
7
                 prevs.append((x,v))
8
            elif world[x][y] == config.IDENT_HUNTER:
9
                 hunters.append((x,y))
10
11
    #numslice: number of possible agents in slice +1
12
    changes = np.zeros((numSlice, 4), dtype=changes_dtype)
13
    changeCounter = 0
14
```

## Parrallel movement calculation

Movement-Calculation in Python; second part

```
for p in preys:
1
       #loop through list and calculate new position
2
        path = helper.calculateSteps(p, world, config.IDENT_PLANT)
3
       if path != None:
4
           changeCounter += 1
5
           newPos = (path[1][0], path[1][1])
6
           changes[changeCounter][0] = p[0] #save old...
7
           changes[changeCounter][1] = p[1]
8
           changes[changeCounter][2] = newPos[0] #and new position in array
q
           changes[changeCounter][3] = newPos[1]
10
   #do the same for hunters
11
   #update function only needs to run on the part of the array that...
12
   changes[0][0] = changeCounter #...actually contains changes
13
   comm.Send(changes, dest=0)
14
```

Our Project

### Problems and Solutions

rank0 Receive

- Problem: Rank0 has to apply all calculated changes
- (Partial) Solution: Already apply changes of fastest process while waiting for callback of the other ranks

```
if rank == 0:
1
        #process changes from other ranks
2
        while received data < numReceives:
3
            changes = np.zeros(**_in respective size_**)
4
            #wait for fastest callback
5
            comm.Recv(changes, source=MPI.ANY_SOURCE, status=status)
6
            #apply changes from data to world
7
            world = sWorld.updateWorld(changes)
8
            received data += 1
9
        sWorld.finishStep()
10
```

- Problem: When the world gets larger, also the broadcasted arrays (world, changes) get larger
- Solution: No solution, it's implicit to the implementation :(
  - -> performance limits?

However: keep the memory size of communicated arrays as small as possible:

```
rank0 Receive
   # smallest possible datatype for changes array (can be further optimized)
1
   if config.WORLD_SIZE * numRows > 255:
2
       changes_dtype = np.uint16
3
       if config.WORLD_SIZE * numRows > 65535:
4
           changes_dtype = np.uint32
5
           if config.WORLD_SIZE * numRows > 4294967295:
6
               changes_dtvpe = np.uint64
7
   else: changes_dtype = np.uint8
8
   #world:10.000 x 10.000; uint8 ~ 95,4MB; changes with n = 50; uint ~7,6MB
q
```

| Our Project | Sequential Approach<br>O | Parallel Approach<br>○○○○○○● | Benchmarking | Conclusion |
|-------------|--------------------------|------------------------------|--------------|------------|
| Does it wo  | ork                      |                              |              |            |

Yes. Populations are as to expect according to the Lotka-Volterra Laws:



# **Base Config**

- World Size=1000, Simulation Steps=100
- Without Paralleling => World Size = 500 , 10 simulation steps => 47 minutes

| Our Project |   | Sequential Approach<br>O | Parallel Approach | Benchmarking<br>○●○○○○ | Conclusion |
|-------------|---|--------------------------|-------------------|------------------------|------------|
| <u>.</u>    | ~ |                          |                   |                        |            |

### Strong Scaling

### Keeping the problem size, increase parallelism



## **Application Performance**

### Cpu Profiling - rank 0



Conclusion

# **Application Performance**

### Visualizing trace - Snakeviz

| simWorld.py:114(InrishBtop)<br>142 s   |   |  |   |  |  |                           |  |
|--|---|--|---|--|--|---------------------------|--|
|  | simWorld.py:63(spawnPlants)<br>115 s  |  |   |  |  |                           | simWorld.py:169(spawnOffspring<br>25.6 s |
|  |   |  | helper.py:28(g<br>1   | jetClosestOfType)<br>102 s   |  |                           | prey.py:11(breed)<br>13.9 s              |
|  |   |  |   |  |  | 1                         |  |
|  |   |  |   |  |  |                           |  |
| ncalls 0   | tottime 🗸   | percall  | 0 cumtime   | percall  |  | filename:lineno(function) | Search:                                  |
| ncalls 0   | tottime 👻   | percall<br>0.07778   | ¢ cumtime<br>186.7  | • percall 0.07778  | ~:0( <method 'mpi4py.mpi.comm'="" 'recv'="" objects="" of="">)</method>  | filename:lineno(function) | Search:                                  |
| ncalls 0<br>400<br>385217  | tottime v<br>186.7<br>102.3   | percall 0.07778 4.29c-05   | • cumtime 186.7 102.3   | <ul> <li>percall</li> <li>0.07778</li> <li>4.29e-05</li> </ul>   | ~30("method 'Reev' of 'mpi4py.MPI.Comm' objects")<br>helper.py:28(getClosestOfType)  | filename:lineno(function) | Search:                                  |
| ncalls 0<br>400<br>385217<br>400   | tottime v<br>186.7<br>102.3<br>60.41  | percall 0.07778 4.29e-05 0.02517   | cumtime<br>186.7<br>102.3<br>99.31  | percall           0.07778           4.29e-05           0.04138   | 0( <method %cev'="" 'mpi4py.mpi.comm'="" objects="" of="">)<br/>helper.py-28(getCloxestOfType)<br/>simWorld.py:96(updateWorld)</method>  | filename:lineno(function) | Search:                                  |
| ncalls ()<br>400<br>385217<br>400<br>947538  | tottime v<br>186.7<br>102.3<br>60.41<br>19.17   | percall 0.07778 4.29c-05 0.02517 6.503c-06   | cumtime 186.7 102.3 99.31 23.52   | <ul> <li>percall</li> <li>0.07778</li> <li>4.29c-05</li> <li>0.04138</li> <li>7.979c-06</li> </ul>   | 0(-anethod Recv' of 'mpi4py.MPLComm' objects-)<br>helper py-28(getClosestOfType)<br>simWorld py:98(updateWorld)<br>simWorld py:12(movePrey)  | flename:lineno(function)  | Search:                                  |
| ncalls ()<br>400<br>385217<br>400<br>947538<br>50  | tottime<br>186.7<br>102.3<br>00.41<br>19.17<br>13.85  | percall 0.07778 4.29e-05 0.02517 6.503e-06 0.1385  | cumtime 186.7 102.3 99.31 23.52 115.2   | <ul> <li>percall</li> <li>0.07778</li> <li>4.29e-05</li> <li>0.04138</li> <li>7.979e-06</li> <li>1.152</li> </ul>  | O(~incthed Recv' of 'unpidpy.MPI.Comm' objects-)<br>helper.py-2idgetClosestO(Type)<br>simWorld.py-9t0qpdateWorld)<br>simWorld.py-121(movePrey)<br>simWorld.py-02(powePlants)   | filename:lineno(function) | Search:                                  |
| ncalls ()<br>1000<br>185217<br>1000<br>147538<br>100<br>157640   | tottime<br>186.7<br>102.3<br>60.41<br>19.17<br>13.85<br>13.57                                       | percall           0.07778           4.29e-05           0.02517           6.503e-06           0.1385           8.71e-06                   | <ul> <li>cuntime</li> <li>186.7</li> <li>102.3</li> <li>99.31</li> <li>23.52</li> <li>115.2</li> <li>15.38</li> </ul>                                   | percall           0.07778           4.29e-05           0.04138           7.979e-06           1.152           9.872e-00   | 30(-method Recy' of 'mpi4py MPLComm' objects-)<br>helper py-24(getClosestOfType)<br>simWorld, py-30(updateWorld)<br>simWorld, py-31(unoveTrey)<br>simWorld, py-35(upoweTsatts)<br>simWorld, py-15(moveTsattsr)   | flename:lineno(function)  | Search:                                  |
| ncalls ()<br>000<br>885217<br>000<br>447538<br>00<br>557640<br>00  | tottime<br>186.7<br>102.3<br>60.41<br>19.17<br>13.85<br>13.57<br>7.346                              | percall           0.07778           4.29c-05           0.02517           0.503c-06           0.1385           8.71c-00           0.07346 | <ul> <li>cuntime</li> <li>186.7</li> <li>102.3</li> <li>99.31</li> <li>23.52</li> <li>115.2</li> <li>15.38</li> <li>25.61</li> </ul>                    | percall           0.07778           4.29c-05           0.04138           7.979c-06           1.152           9.872c-06           0.2561  | 0(-anethod Recv' of 'mpi4py.MPLComm' objects-)<br>helper py-28(gatclosestOfType)<br>simWorld py:06(gataletWorld)<br>simWorld py:05(gawaPlants)<br>simWorld py:05(gawaPlants)<br>simWorld py:15(moveFung)   | filename:linens(function) | Search:                                  |
| ncalls         0           000         0           85217         0           447538         0           57640         0           99856         0  | tottime<br>186.7<br>102.3<br>60.41<br>19.17<br>13.85<br>13.57<br>7.346<br>5.251                     | percall<br>0.07778<br>4.29c-05<br>0.02517<br>0.02517<br>0.1385<br>8.71c-00<br>0.07340<br>0.07340   | <ul> <li>cumtime</li> <li>186.7</li> <li>102.3</li> <li>99.31</li> <li>23.52</li> <li>115.2</li> <li>115.3</li> <li>25.61</li> <li>13.93</li> </ul>     | percell           0.07778           4.29e.05           0.04138           7.979e.06           1.152           9.872e.06           0.2561           1.62e.06                                       | :0(-sincthed Reev' of 'unpidpy.MPI.Comm' objects>)<br>helper.py.28(girtClosestOfType)<br>simWorld.py:121(movePrey)<br>simWorld.py:121(movePrey)<br>simWorld.py:13(movePrey)<br>simWorld.py:13(movePrey)<br>simWorld.py:13(movePrey)<br>simWorld.py:16(pewnOfTpring)<br>prey.py:11(bree)  | filename:Inces(function)  | Search:                                  |
| ncalls         0           00         85217         0           47538         0         0           57640         0         99850           97818         8718         9718                                | tottime    186.7  102.3  60.41  19.17  13.85  13.57  7.346  5.251  3.575                            | percall 0.07778 4.29c-05 0.02517 0.503c-00 0.1385 8.71c-00 0.07340 0.0106c-07 5.511c-07  | euntime 186.7 102.3 99.31 23.52 115.2 15.38 25.61 13.93 8.448   | percall           0.0778           4.2%-05           0.04138           7.979-00           1.152           9.872-06           0.2561           1.302-06   | 0(-snethod 'Recs' of 'mpi4py.MPLComm' objects-)<br>helper py 28(gr6(LonestOT)pe)<br>simWorld, py:36(grandatWorld)<br>simWorld, py:31(movePrey)<br>simWorld, py:31(movePrey)<br>simWorld, py:16(repare States)<br>simWorld, py:11(breed)<br>random py:20(morlmaps)  | flename:linens(function)  | Search:                                  |
| ncalls         0           00         85217         0           00         47538         0           0         57640         0           0         99856         9           87818         87818         9 | tottime ,<br>186,7<br>102,3<br>60,41<br>19,17<br>13,85<br>13,57<br>7,346<br>5,251<br>3,375<br>3,504 | percall 0.07778 4.29c-05 0.02517 0.02517 0.035 0.1385 0.07340 0.07340 0.006-07 5.511c-07 5.5410-07                                       | cumtime           186.7           102.3           99.31           99.32           115.3           25.61           13.33           8.448           4.872 | percall           0.0778           4.290-05           0.04138           7.9790-00           1.152           9.8720-00           0.2561           1.620-00           1.3020-00           7.516-07 | 0(-anethod Recv' of 'mpi4py,MPLComm' objecto-)<br>helper.py-28(getClosestOfType)<br>simWorld, py:05(guwerbey)<br>simWorld, py:05(guwerbey)<br>simWorld, py:05(guwerbist)<br>simWorld, py:05(guwerbist)<br>prey.py:11(hecol)<br>prey.py:11(hecol)<br>prey.py:10(nadhonge)<br>prendmarking, py:200(nadhonge)<br>prendmarking, py:200(nadhonge)<br>prendmarking, py:200(nadhonge)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking)<br>prey.py:10(predmarking) | flename:linens(function)  | Search:                                  |

# **Application Performance**

#### **Other ranks**

| 2<br>helper.py:4(calculateSteps)<br>3.65e+3.s |  |
|---|--|
| helper.py-4(calculateSteps)<br>3.65e+3 s      |  |
| 8   |  |
| helper.py.4(calculateSteps)<br>503 s          |  |
| helper.py:4(calculateSteps)<br>503 s          |  |
|   |  |
| helper.py.4(calculateSteps)<br>237 s          |  |
| helper.py.4(calculateSteps)<br>237 s          |  |
|   |  |
| A helper.py.4(calculateSteps)<br>169 s        |  |
| helper.py.4(calculateSteps)<br>169 s          |  |

| Our Project | Sequential Approach<br>O | Parallel Approach | Benchmarking<br>○○○○○● | Conclusion |
|-------------|--------------------------|-------------------|------------------------|------------|
|             |                          |                   |                        |            |

# Weak Scaling

### Increase the problem size with parallelism



Kimia Taba and Vincent Hasse

University of Göttingen

# **Future Work**

- Writing report
- Memory Profiling
- Change path finding algorithm

# Conclusion

- Goal: Simulating a predator-prey relationship
- Achieve lots of Performance improvements by using mpi