A Framework for distributing Agent-based simulations

Gennaro Cordasco, Rosario De Chiara, Ada Mancuso, Dario Mazzeo, Vittorio Scarano and Carmine Spagnuolo

Outline

- Agent Based Simulations (ABMs)
- MASON
- Distributed ABMs
- DMASON
  - issues
  - architecture
- MASON vs DMASON
- Tests

Agent-based simulation

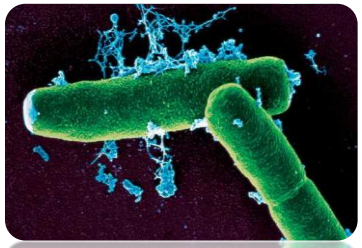
- Simulate the actions and interactions of autonomous individual agents with a view to assessing their effects on the system as a whole
  - Investigated since the 1980s
  - Early works take inspiration from particles [R83]
- Applications
  - Biology, economics, sociology …
- Why parallel?
  - Huge number of agents
  - Complex behaviours

Motivation

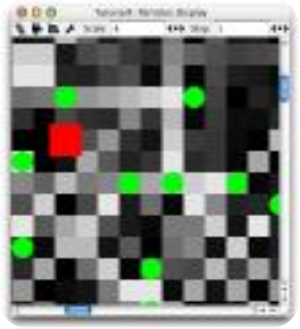
Social Science

Economy Science

Biology



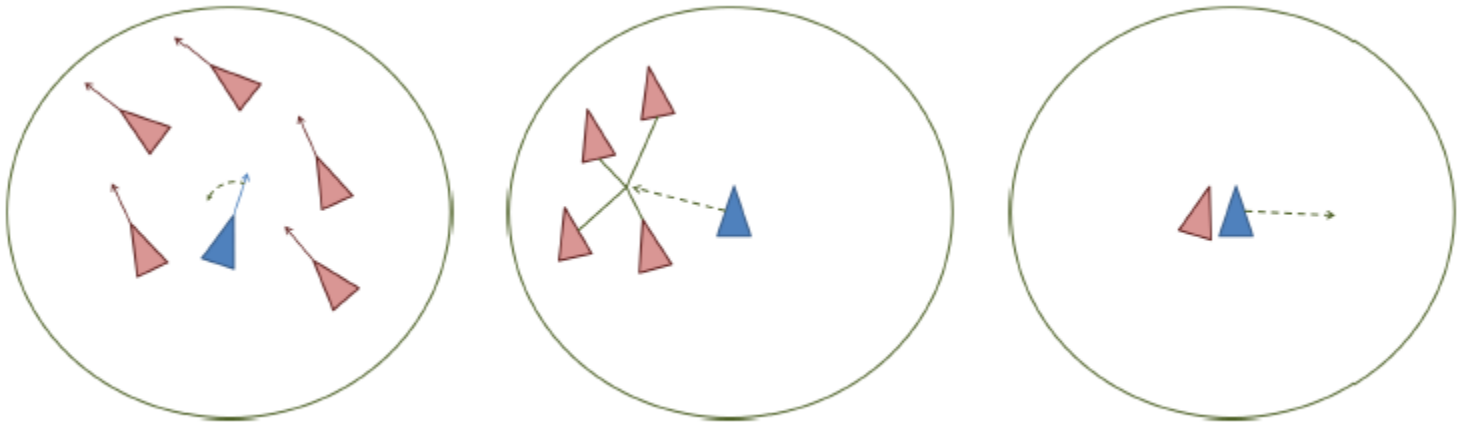
Fisics



Artificial Intelligence

An Example: Reynolds' Model

- *Alignment*: steer towards the average heading of local flock-mates
- *Cohesion*: steer to move toward the average position of local flock-mates
- *Separation*: steer to avoid crowding local flock-mates



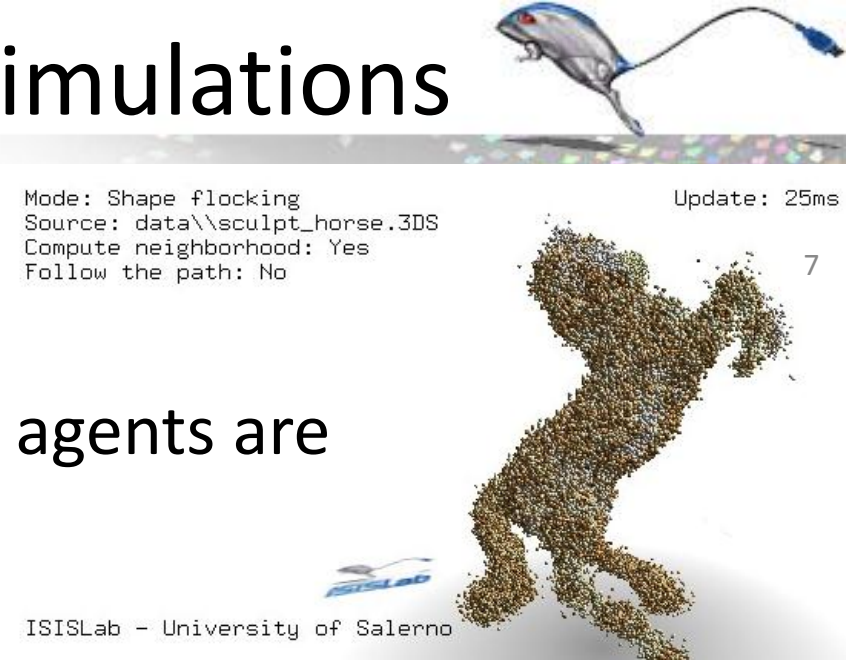
Alignment **Cohesion** **Separation**

Features of AB Simulation platforms

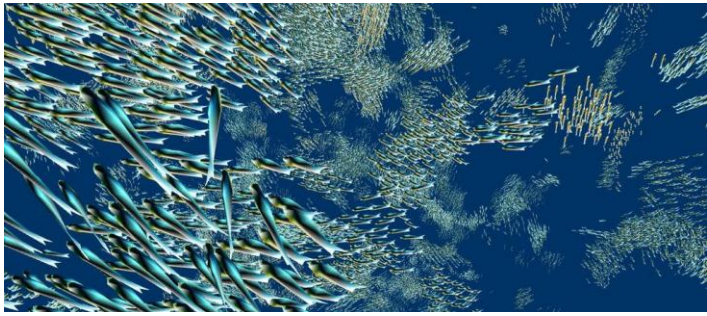
ISISLab

- Flexibility
  - customization of agents and their behaviors
- Speed and efficiency
  - scalable beyond millions, hundred of millions, billions
  - batch processing
  - interactive simulation rate (in the order of seconds)
- Testing and validation suites
- Research facilities (logging, graphing, etc.)
- Open-source
  - communities of public/private entities, users, developers, researchers are crucial to ensure long-term sustainability and deep impact on current research practices

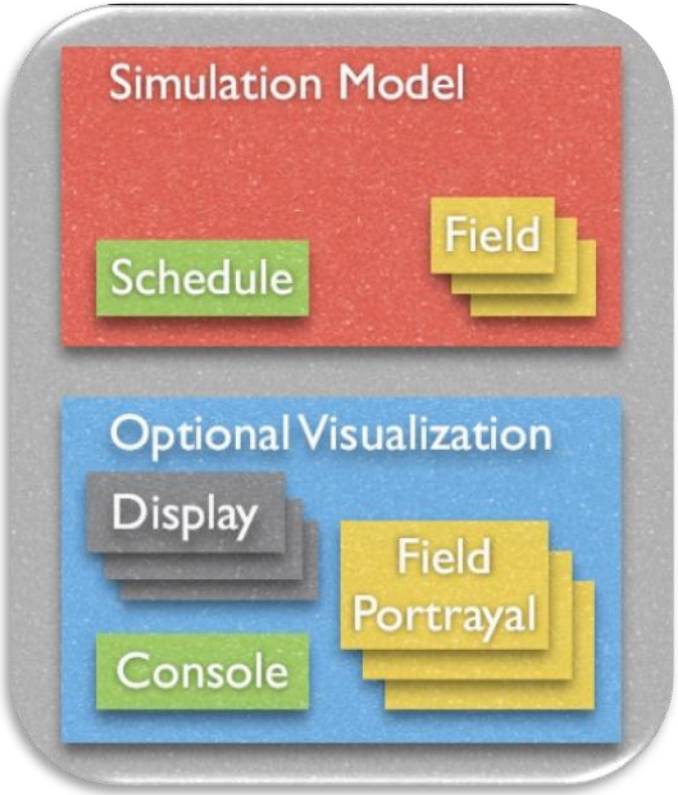
Massive Agent-Based Simulations



- *Massive*: when the simulated agents are
  - extremely numerous,
  - complex to simulate
  - with non-linear, dynamic behavior
- A Short Answer: "smart" parallelization

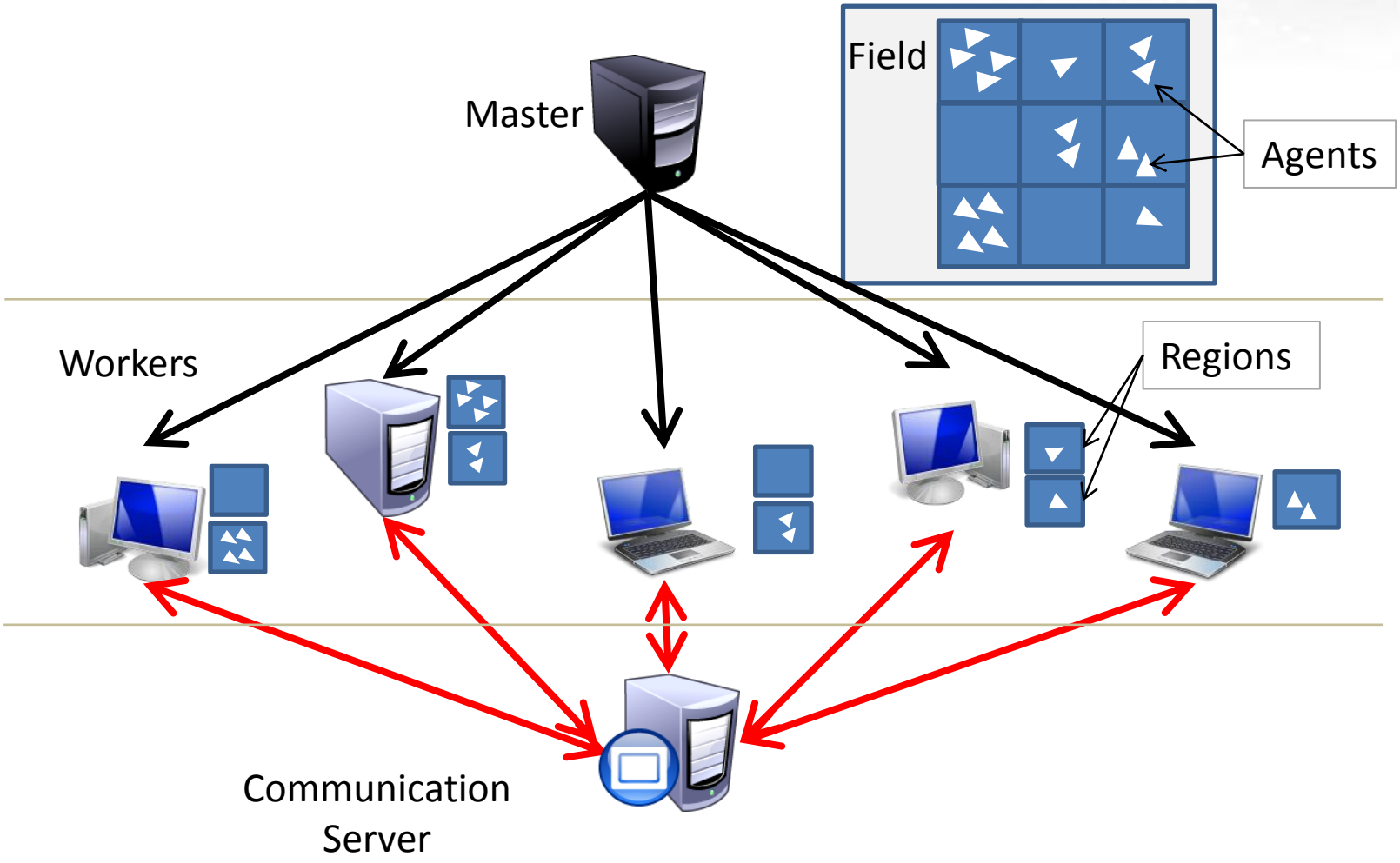


Why MASON ?



- Mason is recognized to be expressive and efficient
- Mason structure: clear separation between Simulation and Visualization
- Back compatibility with simulations already present in the framework

DMASON


Field
Agents
Master
Workers
Regions
Communication
Server

DMASON Issues

ISISLab

Work Partitioning

Synchronization

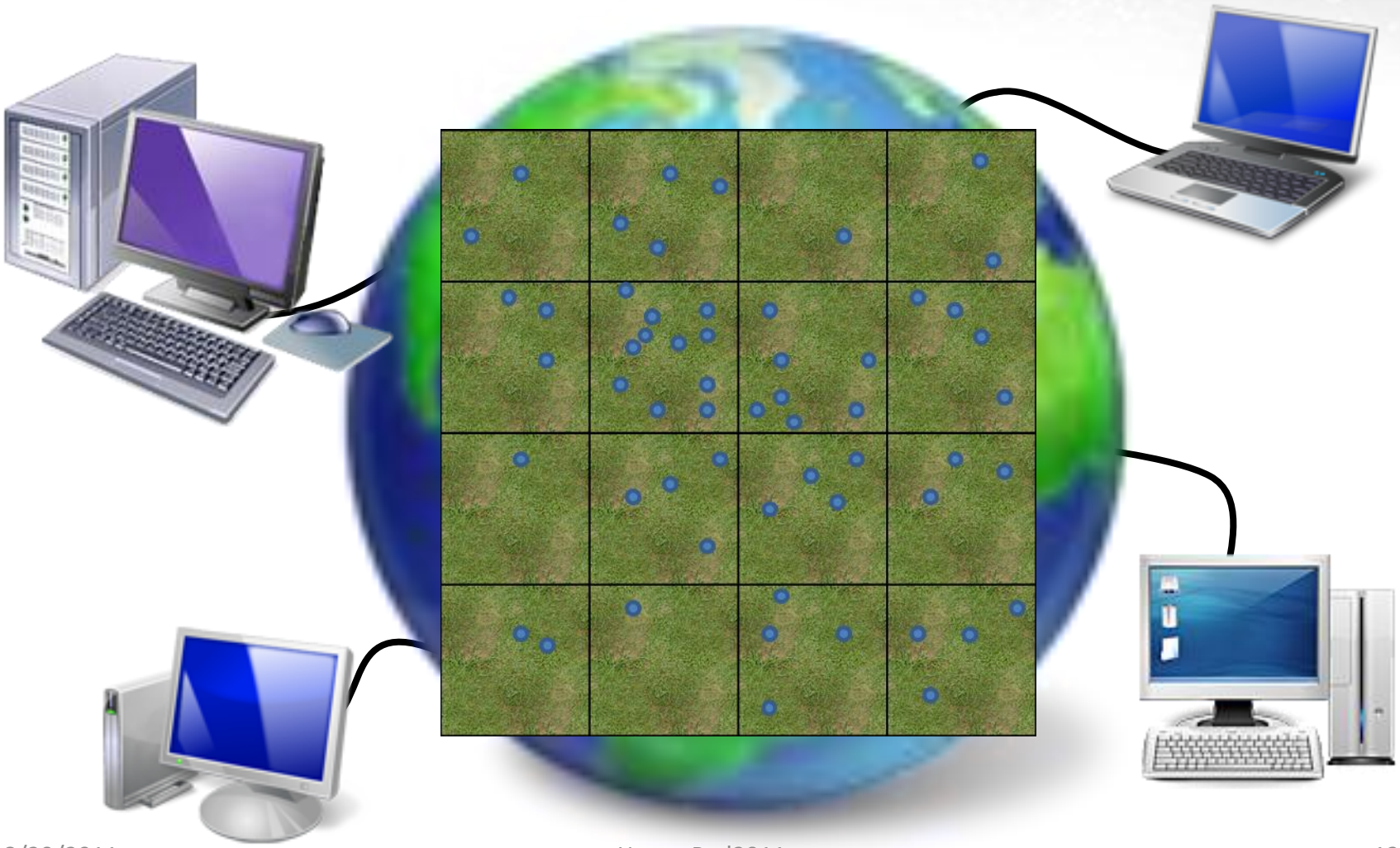
Communication

Reproducibility

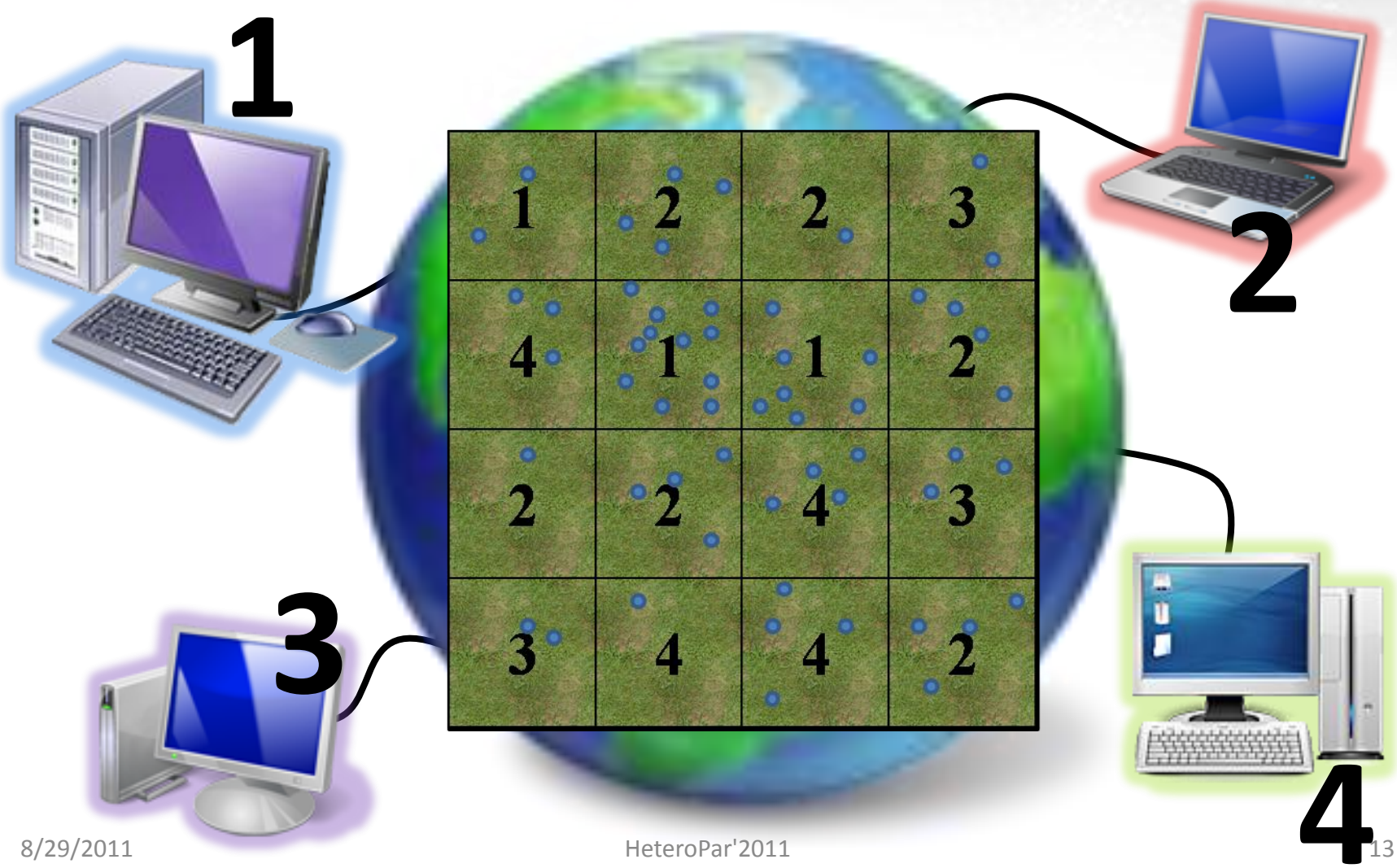
Agents vs Space Partitioning

- Agents Partitioning assigns a fixed number of agents to each available worker
  - Self balanced
  - Requires an all to all communication
- Space Partitioning partitions the simulation space into regions. Each region is assigned to a worker which is in charge of simulating all the agents belonging to the region
  - A small amount of communication is required
  - Agents can migrate
  - Load balancing is not guaranteed

DMASON: Field Partitioning



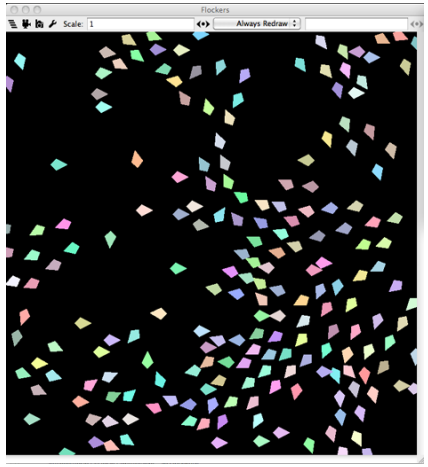
DMASON: Field Partitioning



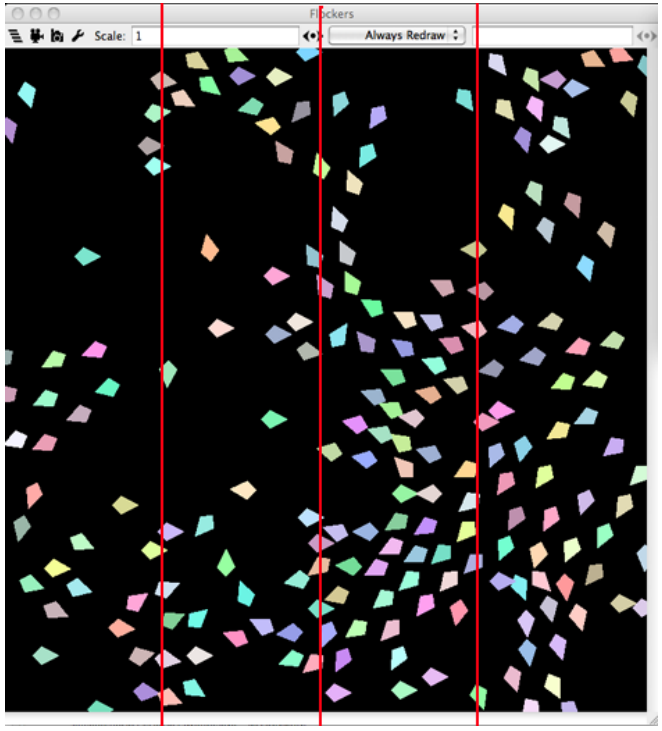
DMASON: Field Partitioning

- DMASON allows to partition the field into regions
- Neighboring regions communicate before each simulation step

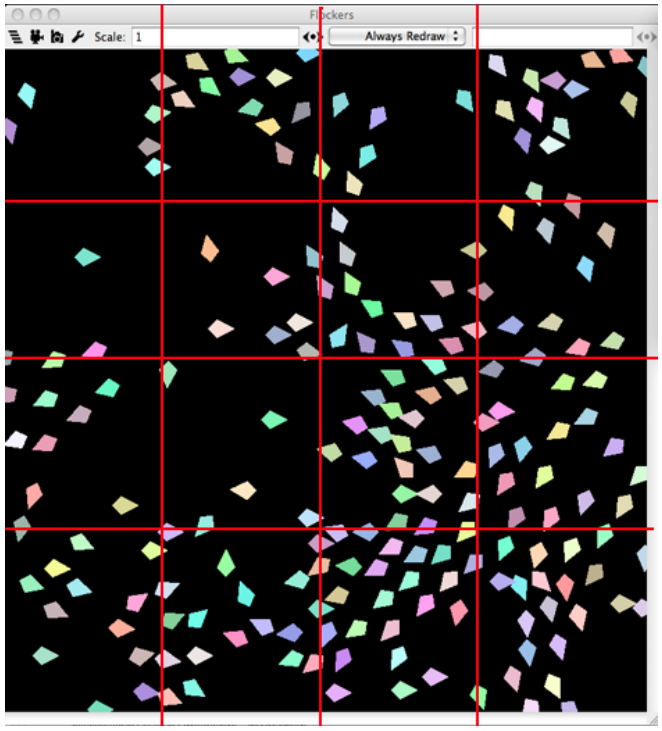
MASON



DMASON 1D

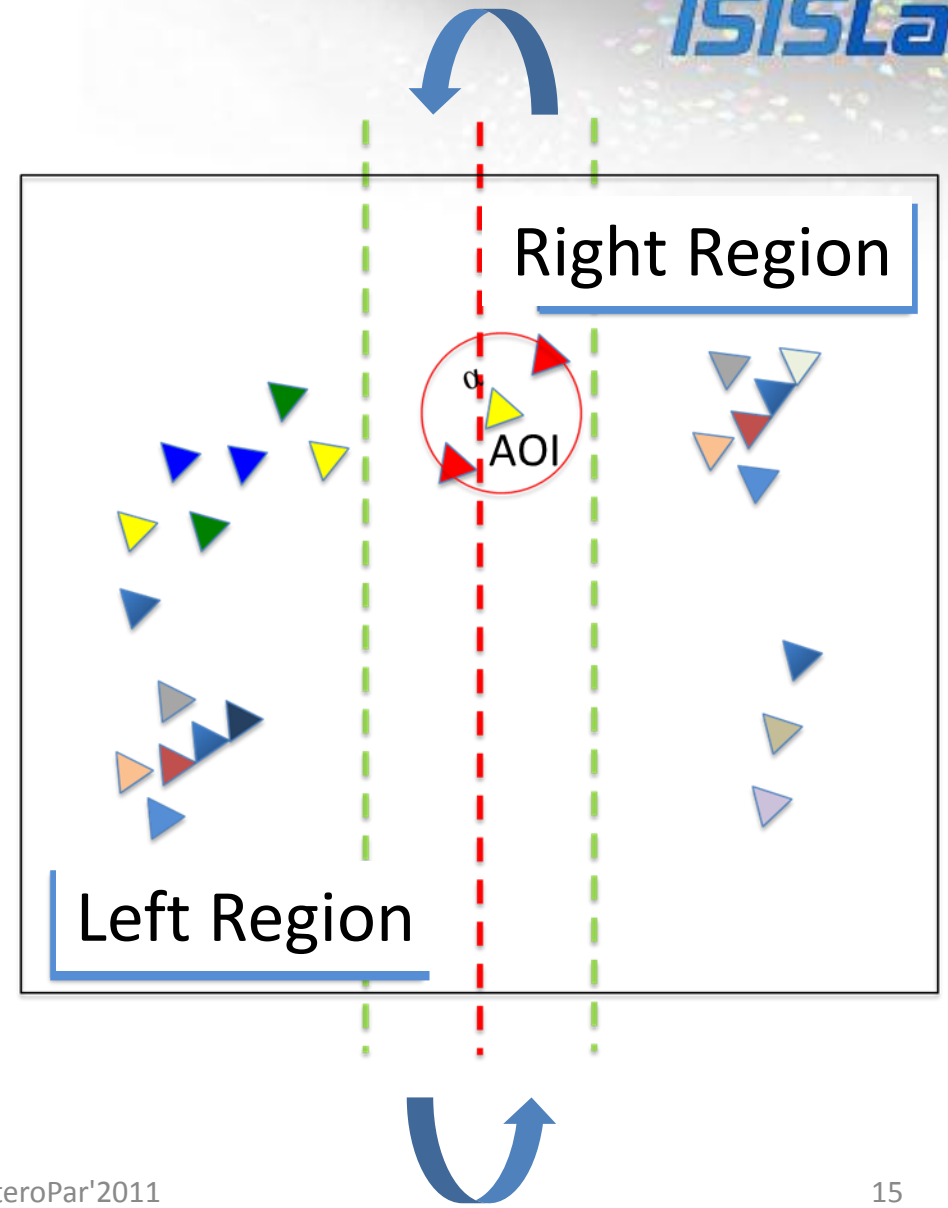


DMASON 2D



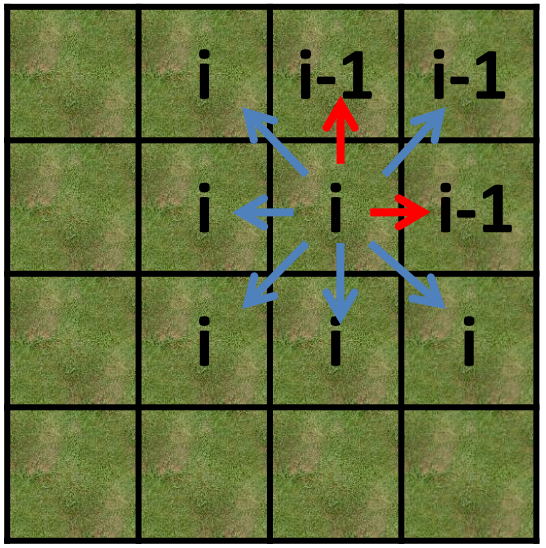
DMASON: Field Partitioning

- A portion of each region is exchanged between neighbor workers before each simulation step
- The size of this portion depends on agents area of interest (AOI)



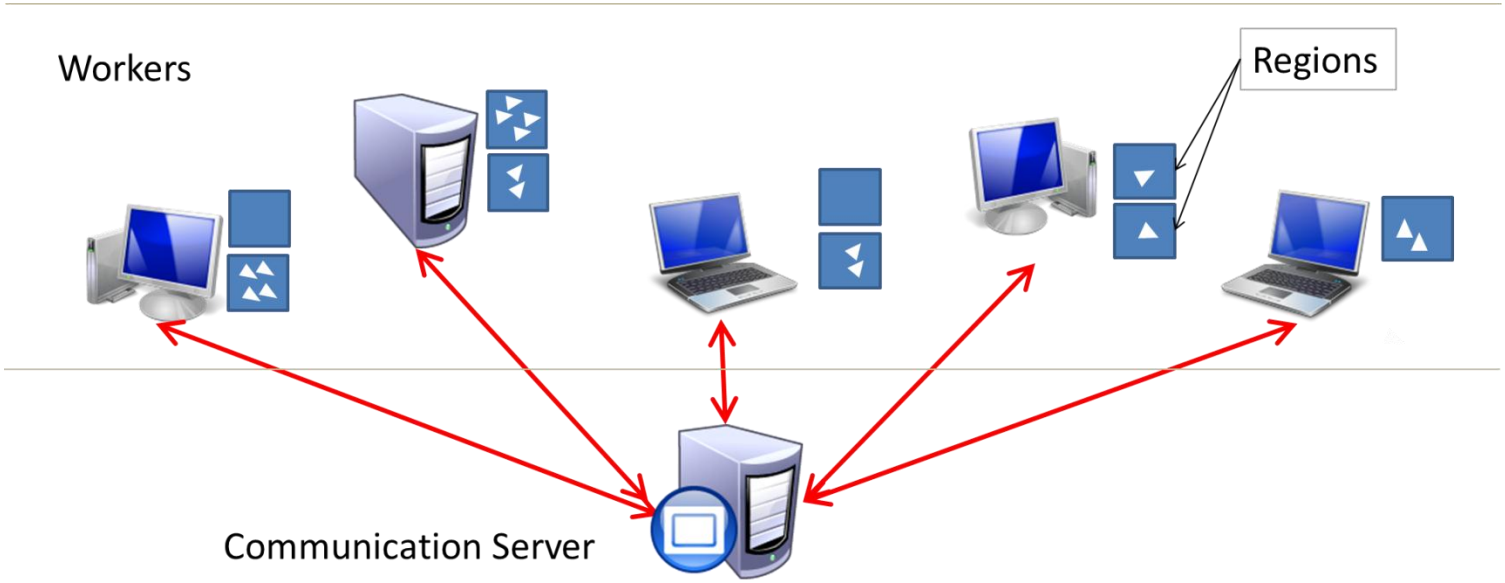
DMASON: Synchronization

- Local synchronization:
  - The step i of region r is computed by using the states i − 1 of r's neighborhood
  - The step i of a region cannot be executed until the states i − 1 of its neighborhood have been computed and delivered.
- No central coordinator
- Simulation speed ≈ slowest region speed



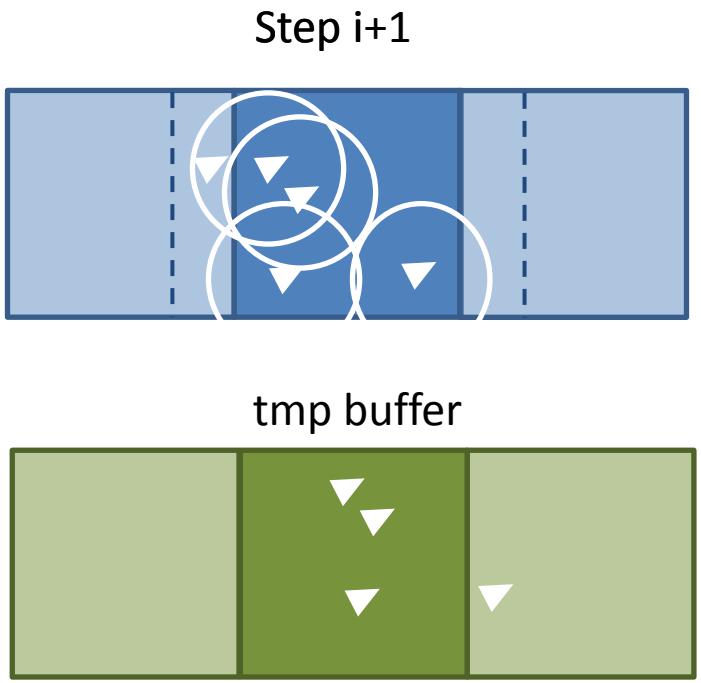
DMASON: Communication

- DMASON uses the publish–subscribe design pattern to propagate agents state information
  - Current version of DMASON uses Java Message Service (JMS) for communication between workers



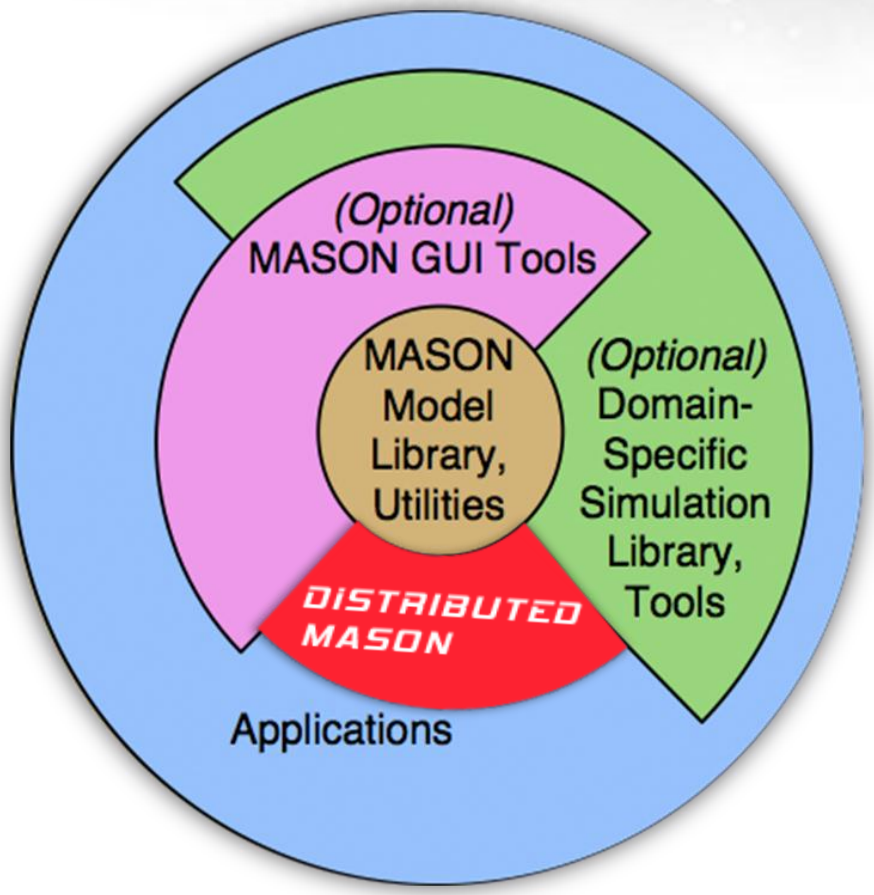
DMASON: Reproducibility

- Agents evolve simultaneously
  - each simulation step can be executed in parallel overall the agents
  - the order in which agents are scheduled does not affect the reproducibility of results
  - neighbors' updates are always processed in the same order



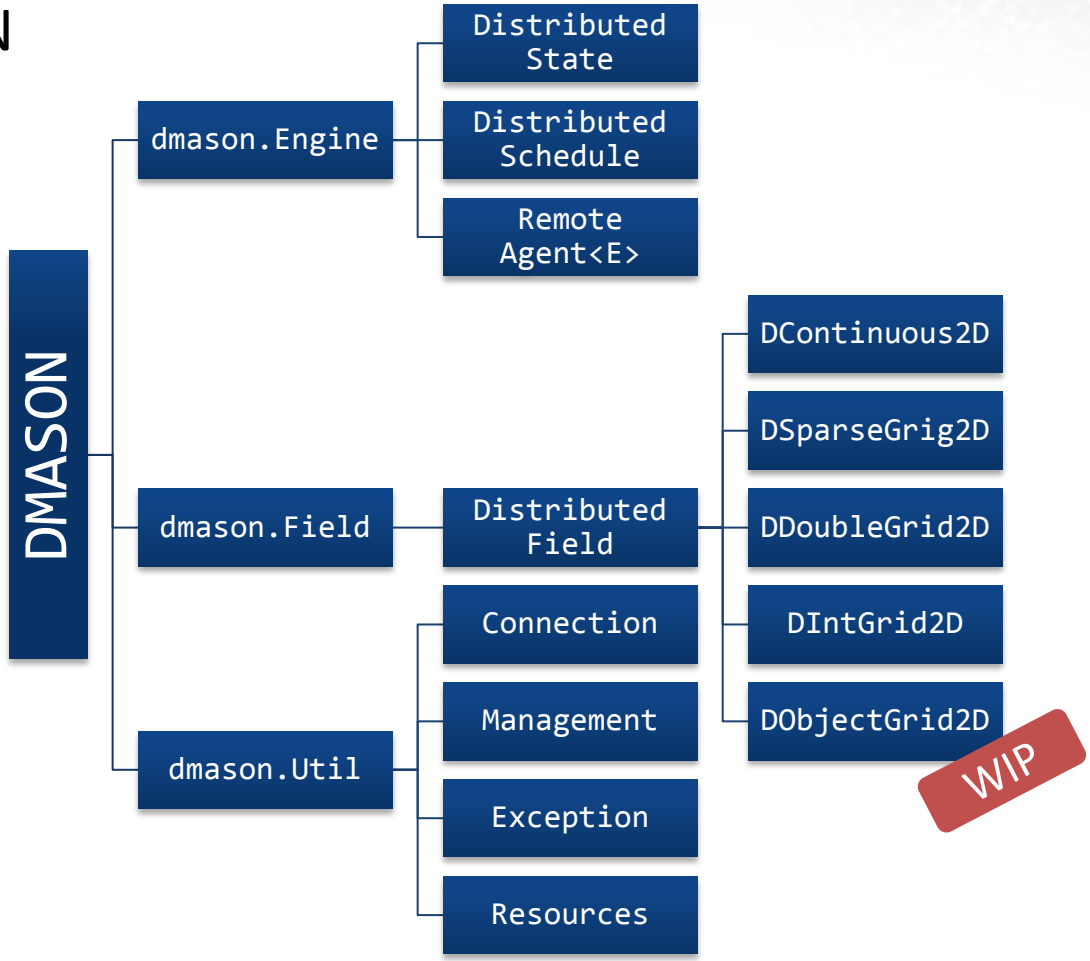
DMASON Architecture

- DMASON is a new layer which extends the MASON simulation layer.
- The new layer does not alter in any way existing layers

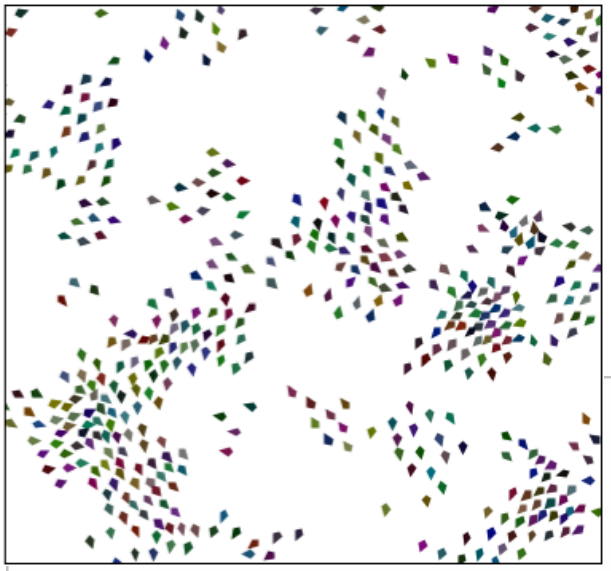


DMASON Architecture

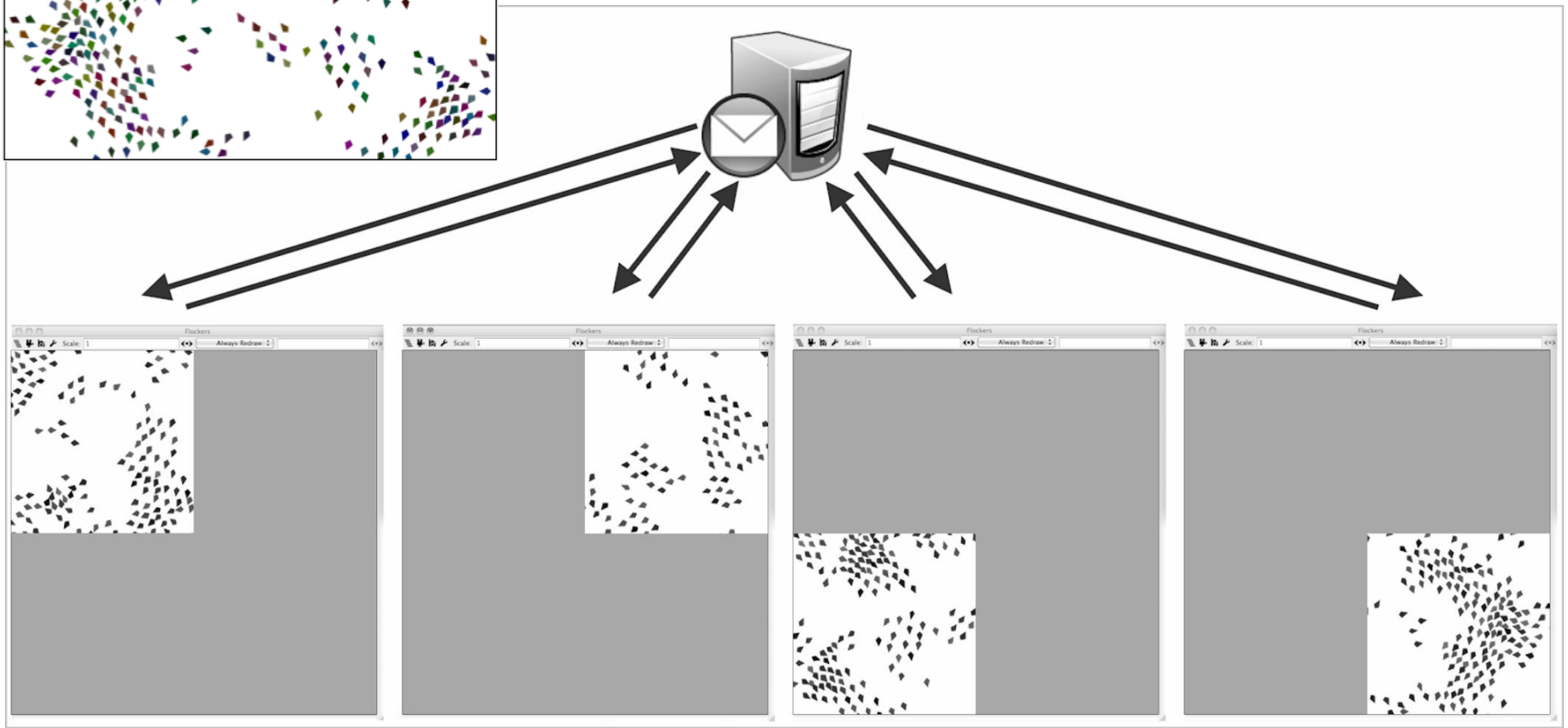
The same structure as MASON



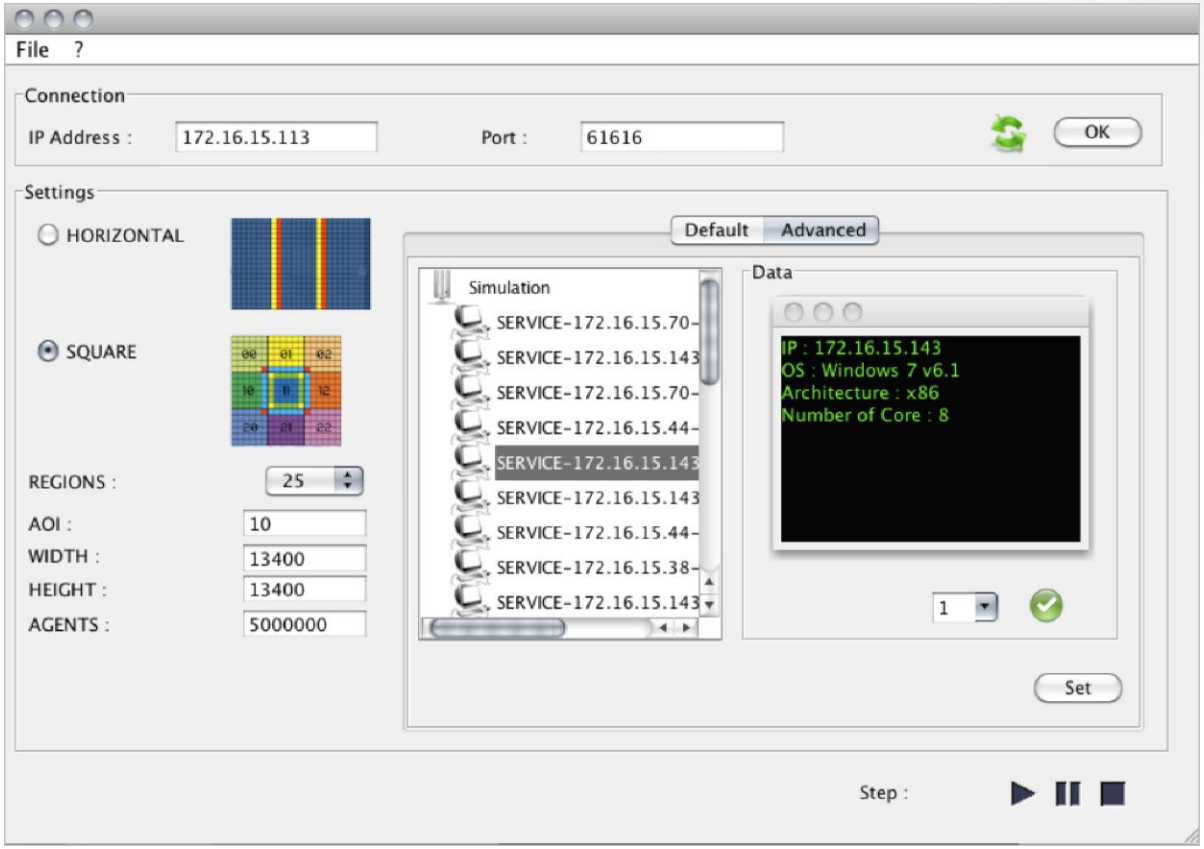
MASON vs DMASON



DFlockers



DMASON: System Management



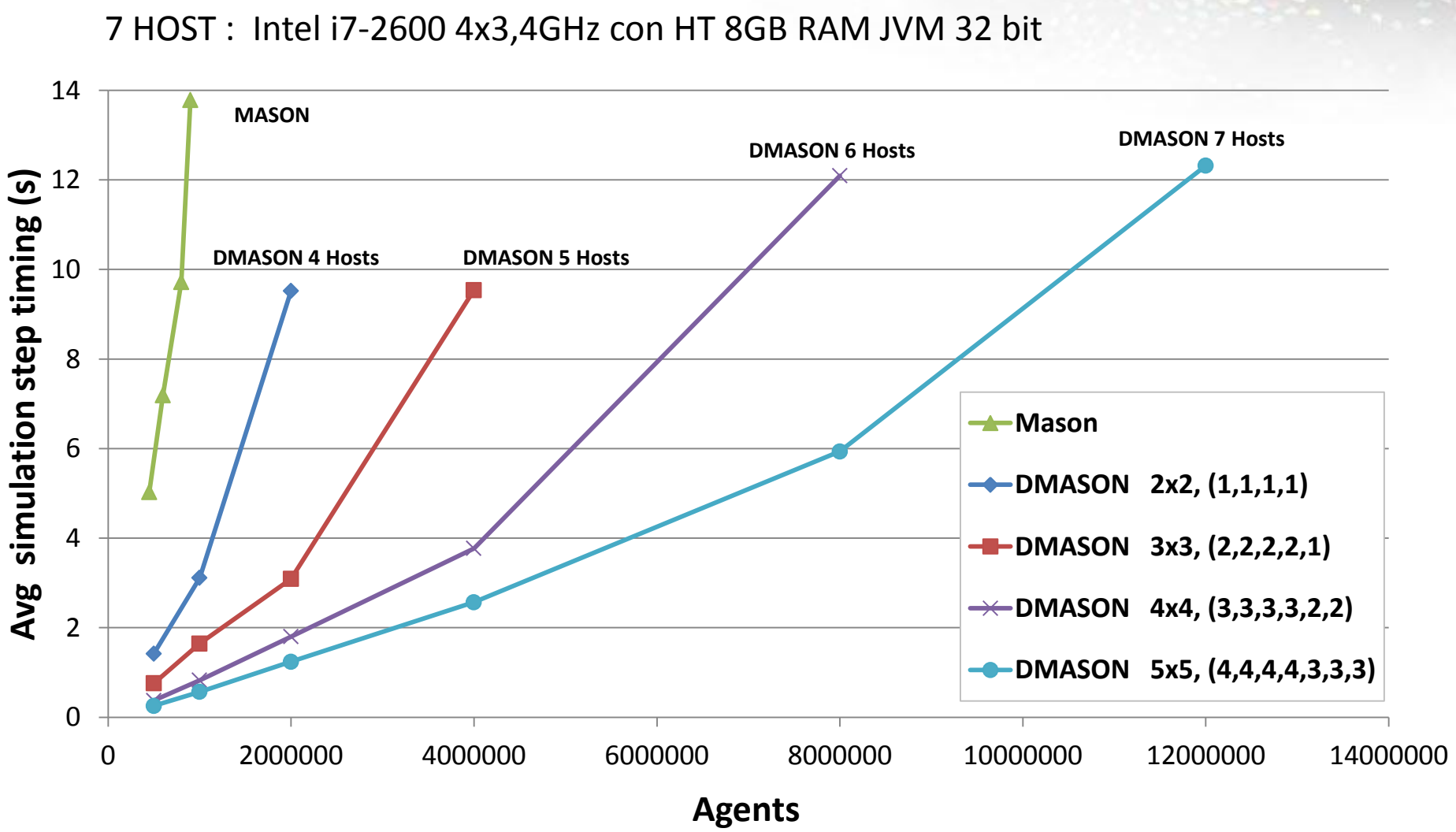
Master Console

Worker

Testing DMASON

- We want to assess that DMASON…
  - …is able to run simulations that are impractical or impossible to execute with MASON
  - …is scalable
  - …allows to develop reproducible simulations (independently from the number of workers)
  - …allows to exploit multicore CPU
  - …allows to exploit heterogeneous hardware

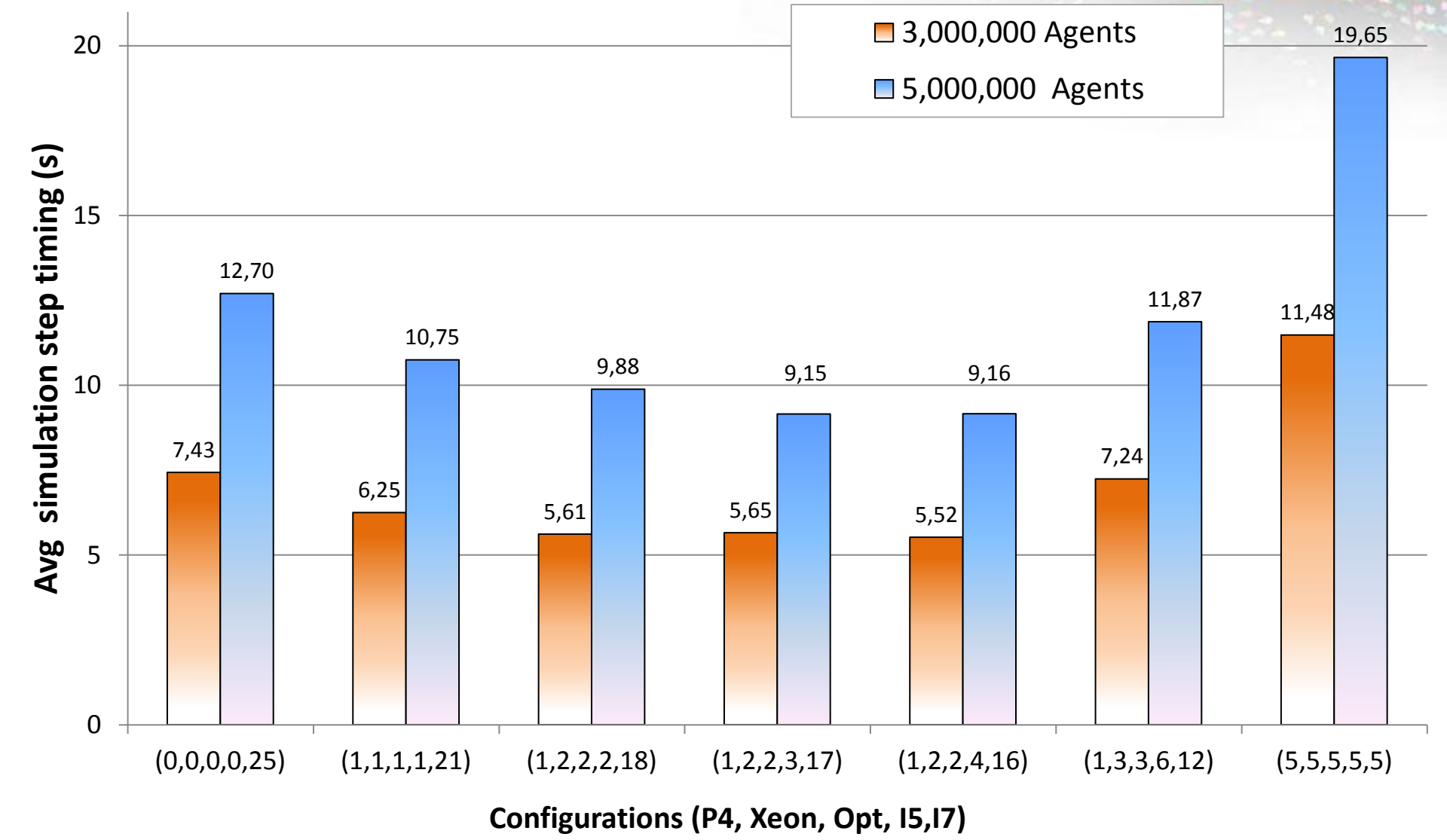
DMASON: homogeneous hosts



DMASON: heterogeneous hosts

Host	Hardware
Host 1	2xAMD Opteron 252-4GB RAM-Win 7 64bit
Host 2	i5 2x2,53 4T.-4GB RAM-Win7 64bit
Host 3	Pentium 4 3,4- 2 Gb RAM-Win XP 32bit
Host 4	2xXeon 2,67-3GB RAM-Win7 32 bit
Host 5	i7-2600 4x3,4 8T.-8GB RAM-Win7 64bit
Server	**Server : V.M. 2xXeon 2.39 -Ram 8 gb**

DMASON: heterogeneous hosts

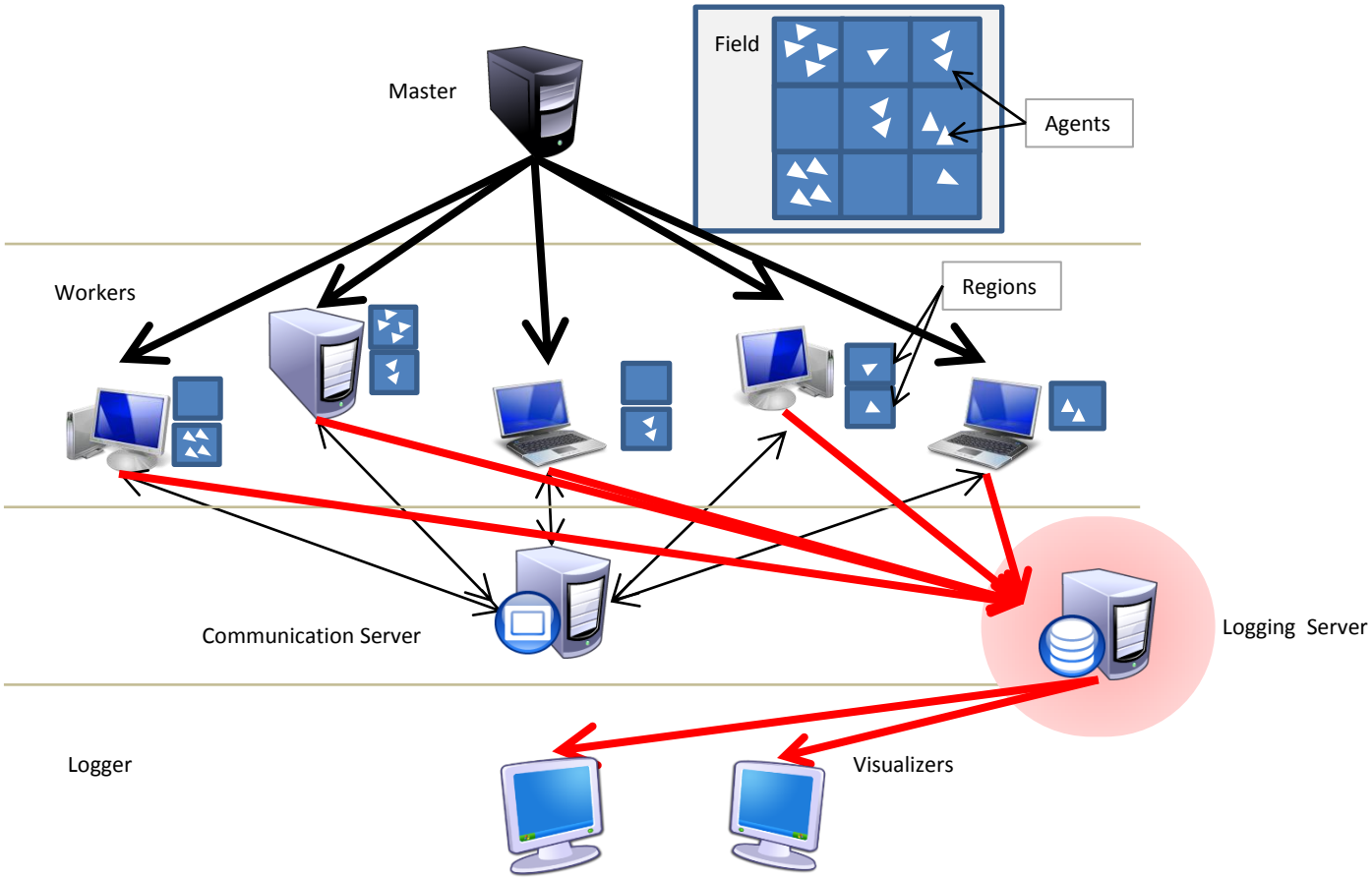


Conclusion

- ABMs are CPU intensive applications and requires large amount of memory
- The need for complex simulations involving a large number of agents is always felt by researchers and practitioners
- DMASON allows to achieve those requirements by harvesting the unused CPU power and Memory

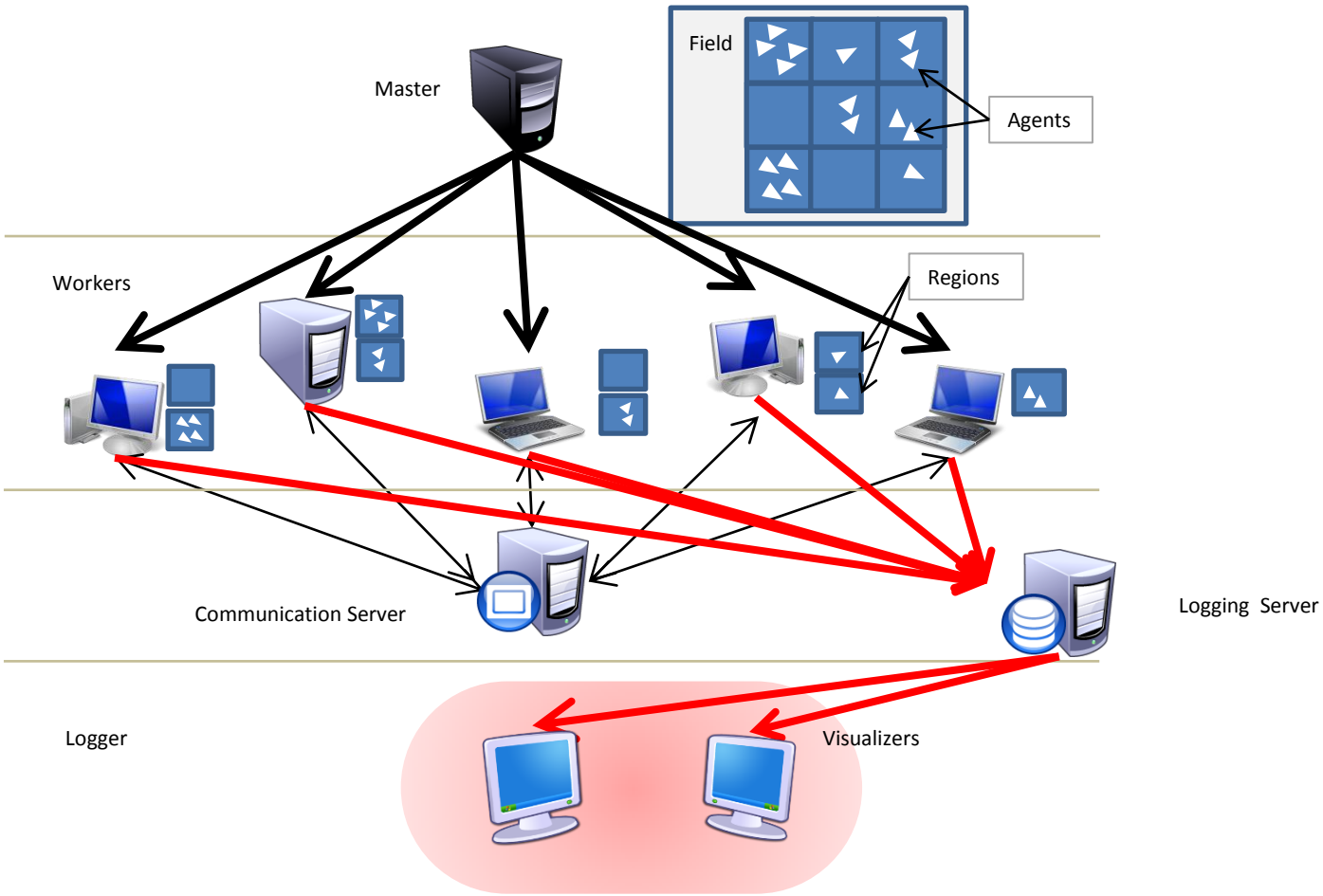
Current work

- Simulation logging and replay



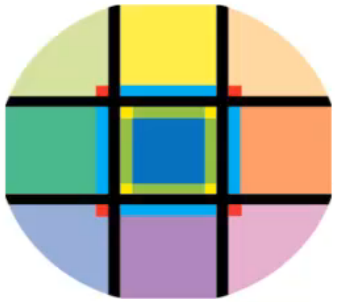
Current work

- Visualization



Future Development

- Development of other distributed fields
- Dynamic Load Balancing
- Distributed communication



D.MASON

Example application Flockers
Square division 4 cells.

Thanks for your attention

Gennaro Cordasco, Rosario De Chiara, Ada Mancuso, Dario Mazzeo, Vittorio Scarano and Carmine Spagnuolo

http://www.isislab.it/projects/dmason/