Empowering Computational Science through Extreme Scalability

Ph.D Candidate: Matteo D'Auria

Supervisor: Vittorio Scarano

University of Salerno
Department of Computer Science
Department of Information and Electrical Engineering and Applied Mathematics

May 14, 2021

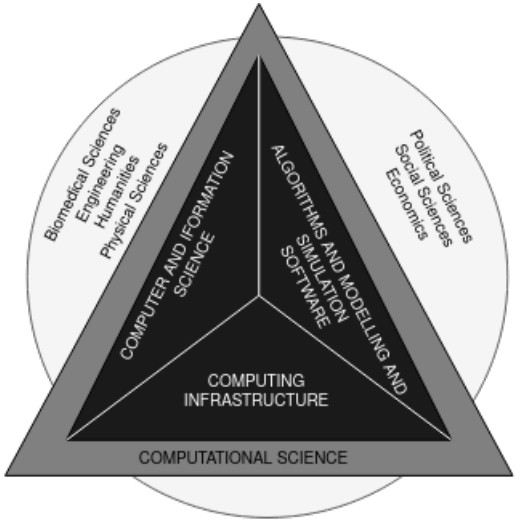
La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005),
Fondo Sociale Europeo, Azione I.1 "Dottorati Innovativi con caratterizzazione Industriale"

Table of Contents

Table of Contents

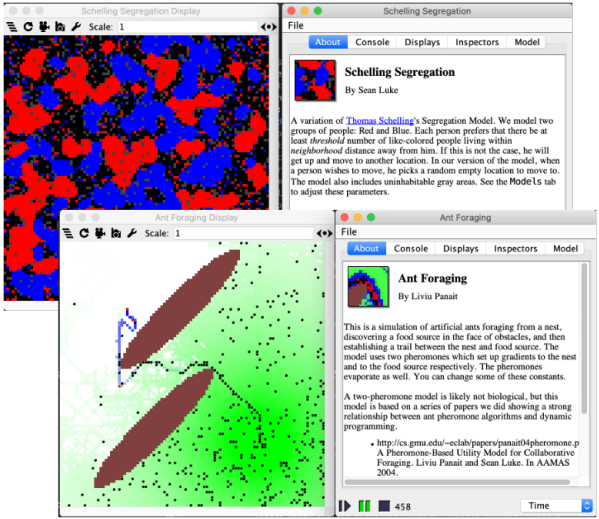
Computational Science

- Multi-Disciplinary.
- Impact on several Sciences.
- Objective: study complex artificial and natural systems using modern computational models.



Agent-Based Models (ABM)

- Necessary given the difficulty or impossibility of reproducing complex systems in the laboratory.
- Bottom-up approach.
- They models the actions and interactions of autonomous agents to assess their effects on the system as a whole.

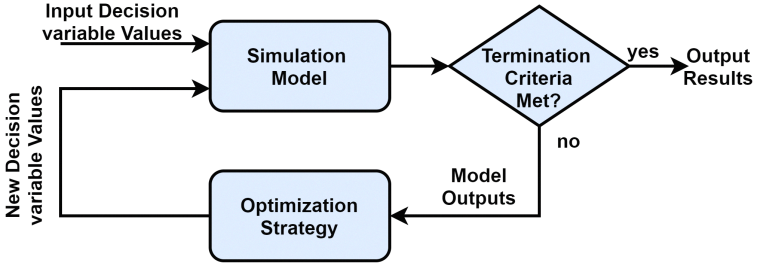


Agent-Based Models

- Useful in observing the behavior of the system as the model parameters change (What-if Scenario).
- Problem: Finding the best simulation parameters value configuration for a particular case study.

Optimization via Simulation (OvS)

- Optimizes the values of the parameters without exploring the entire space of possible solutions.
- Minimizes the use of resources spent by maximizing the information obtained from a simulation.



The need of Scalability

Observation 1:

These tools require high computational resources accessible with the use of distributed and parallel computing techniques.

Observation 2:

The notion of scalability becomes essential.

Definition:

An application is scalable when it can efficiently exploit a growing computational power.

The point of view of Computational Scientists

Observation 1:

Implementing scalable solutions requires a deep understanding of parallel and distributed computing architectures and paradigms.

Observation 2:

The cultural background of computational scientists does not have these skills.

Problem:

They spend most of their time making applications scalable by taking time away from implementing the application logic.

Dissertation contributions

- Simplify the work of domain experts by providing languages and tools to allow them to write effective and efficient applications taking full advantage of scalability offered by modern computing techniques.

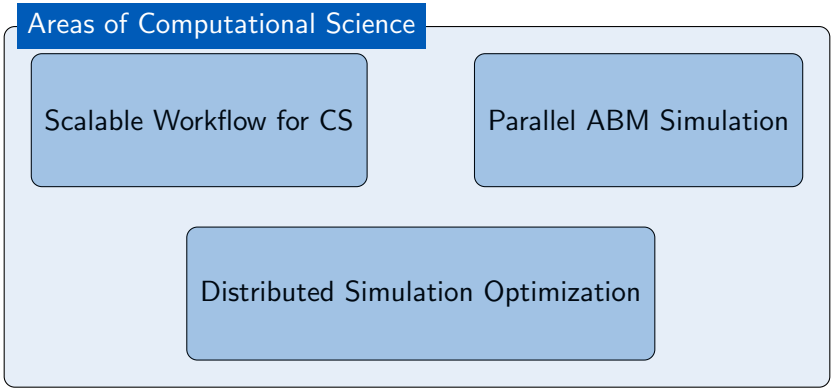


Table of Contents

FLY Language motivation

- It is possible to create scientific workflows in different languages (Fortran, R, Python, etc.).
- Writing applications that exploit parallel or distributed computing requires advanced skills (not common for domain experts).
- From this was born the idea of creating a language that:
  - Natively supports parallel constructs (fork/join, map/reduce, etc.).
  - Provides a programming model compatible with distributed computing.
  - Support the use of libraries made in languages supported by the cloud (NodeJs, Python, etc.).
  - Abstract the interaction with the computing platform used.

FLY Language [1] [2]

- Domain-Specific Language for Computational Science on Cloud computing platforms.
- FLY perceives a Cloud computing infrastructure as a parallel computing architecture on which it possible to run parts of the execution flow.
- Leverage on the Cloud Function-as-a-Service paradigm (also called Serverless Computing).

[1] Cordasco, G.; D'Auria, M.; Negro, A.; Scarano, V. & Spagnuolo, C., "FLY: A Domain-Specific Language for Scientific Computing on FaaS", Lecture Notes in Computer Science, Springer, 2020. 25th International European Conference on Parallel and Distributed Computing, EuroPar 2019.

[2] Cordasco, G.; D'Auria, M.; Negro, A.; Scarano, V. & Spagnuolo, C., "Toward a domain-specific language for scientific workflow-based applications on multicloud system", Concurrency and Computation: Practice and Experience, John Wiley and Sons Ltd, 2020.

FLY Language

- Expressiveness: in the design of scientific workflows.
- Usability: writing FLY programs must be simple for domain experts (interaction with the Cloud environment must be completely transparent).
- Scalability: both on symmetric multi-processing architectures (SMP) and Cloud computing infrastructures.

FLY Function

- The crucial concept of the FLY Language.
- An independent block of code that can be executed concurrently.
- The language provides constructs for:
  - The definition, execution, and synchronization of FLY functions.
  - The communication between FLY functions.

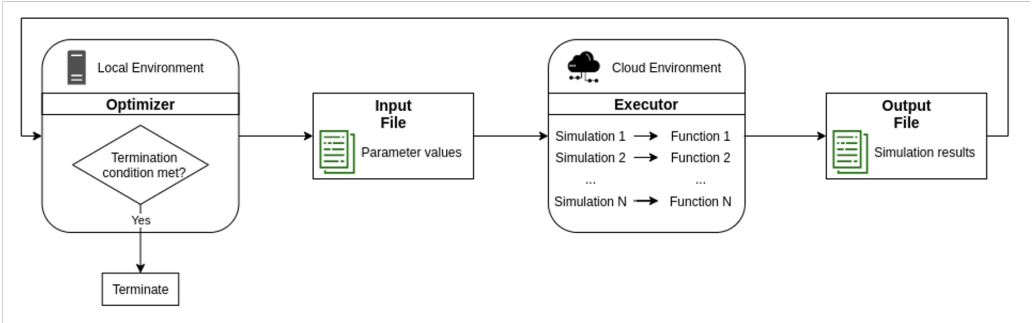
FLY Language: Performance and Cost Evaluation on Word Count

- Comparison between a FLY program (run using different computing environments) and a MapReduce program using an Amazon Elastic MapReduce cluster of 16 nodes (128 total vCPUs).

Configurations			FLY Java		SMP-4			SMP-64			AWS-64			H-16		
MB	N	Tot. GB	T (h)	C ($)	T(h)	S	C ($)	T(h)	S	C ($)	T(h)	S	C ($)	T (h)	S	C ($)
125	500	$\sim$ 62	0.751	0.2	0.324	2.3	**0.1**	0.016	44.9	0.1	**0.015**	**47.7**	0.2	0.061	12.3	6.6
125	1500	$\sim$ 187	3.036	0.6	1.042	2.9	**0.2**	0.089	34.1	0.3	**0.018**	**166.6**	0.5	0.103	29.5	7.3
125	2000	$\sim$ 250	3.711	0.7	1.351	2.7	**0.3**	0.124	30.0	0.4	**0.018**	**196.1**	1.0	0.106	35.0	7.3
250	500	$\sim$ 125	1.518	0.3	0.670	2.3	**0.1**	0.062	24.3	0.2	**0.018**	**82.9**	0.5	0.560	27.1	6.5
250	1500	$\sim$ 375	4.659	0.9	2.069	2.3	**0.4**	0.177	26.3	0.6	**0.023**	**198.6**	1.5	0.146	29.9	8.1
250	2000	$\sim$ 500	6.060	1.2	2.738	2.2	**0.5**	0.234	26.0	0.8	**0.024**	**250.6**	2.0	0.205	29.6	8.9
500	500	$\sim$ 250	2.911	0.6	1.323	2.2	**0.3**	0.116	25.0	0.4	**0.023**	**121.8**	1.0	0.106	27.5	7.3
500	1500	$\sim$ 750	9.017	1.8	3.981	2.3	**0.8**	0.359	25.1	1.2	**0.034**	**261.3**	2.5	0.302	29.9	10.6
500	2000	$\sim$ 1000	10.918	2.2	5.065	2.2	**1.1**	0.437	25.0	1.4	**0.035**	**307.6**	3.8	0.400	27.3	12.2

FLY Language - Serverless Simulation Optimization

- Is it possible to use the FaaS paradigm to speed up the OvS process?



Serverless Simulation Optimization: Customer Allocation Problem (CA)

- Product distribution network planning.
- Objective: minimizing the total logistic cost, expressed as the sum of the costs of activation and deactivation, storage, transport, and customer service.
- Solution: The definition of the customer nodes that each production or distribution node must serve.
- Commonly divided into several instances of Capacitated Vehicle Routing Problem (CVRP).

Serverless Simulation Optimization: Customer Allocation Problem

- OPTNet: manages the OvS process, generates different configurations of the CA problem in each round, and divides them into CVRP instances.
- FLY: concurrently solves CVRP instances.

Test run on a 64-core machine with 1000 concurrent functions:

- Speedup> x5
- Costs> x4

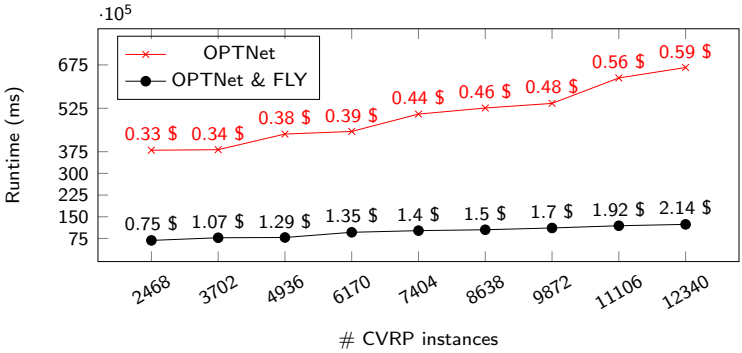


Table of Contents

Yet another ABM simulation library?

- The success of ABM simulations has led to the need to improve their performance constantly.
- Several distributed frameworks have been implemented, but they have problems of efficiency in accessing global data.
- In these cases, parallel or sequential ABM simulations provide better performance.
- High-performance ABM simulations are built on performance-critical operations and show multiple levels of concurrency.
- Rust is a multi-paradigm programming language with comparable performance to C with both a memory and thread-safe model.

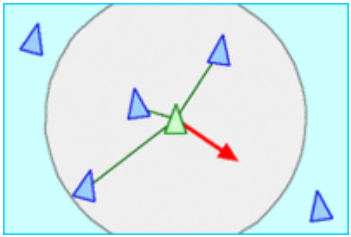
Rust-AB [3]

- A Discrete Event Simulation library written in the Rust language.
- Design Pattern: Model-View-Controller.
- Released open-source and available on Github.
- Two execution modes: sequential and parallel.
- Limitations:
  - The current version of Rust-AB does not support multiple agent definitions.
  - The field can only contain objects of the same type.

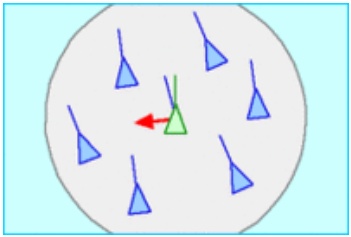
Rust-AB

[3] Antelmi, A.; Cordasco, G.; D'Auria, M.; De Vinco, D.; Negro, A. & Spagnuolo, C., "On Evaluating Rust as a Programming Language for the Future of Massive Agent-Based Simulations", Communications in Computer and Information Science, Springer, 2019, 19th Asia Simulation Conference, AsiaSim 2019.

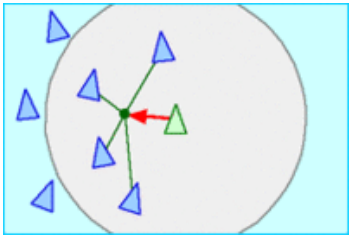
Rust-AB: Boids Model by Craig Reynolds, 1986



(a) Separation: steering to avoid crowding neighboring flockmates.



(b) Alignment: steering towards the average direction of nearby flockmates.



(c) Cohesion: steering to move towards the average position (center of mass) of nearby flockmates.

Rust-AB: Strong Scalability Analisys

- Each test performs 50 simulation steps varying the number of threads.
- EC2 c5.24xlarge machine (96 vCPU, 192 GB of RAM and 25 Gb/s of bandwidth).

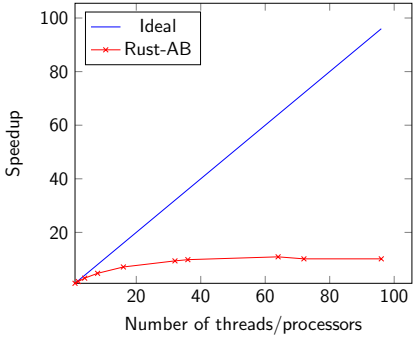


Figure: Boids Model with 500.000 agents.

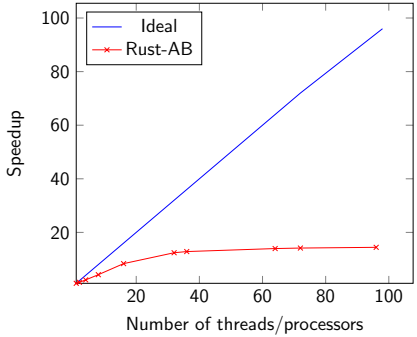


Figure: Boids Model with 1.000.000 agents.

Table of Contents

Distributed Simulation Optimization

- Three tools are developed for exploiting parallel and distributed systems' computational power to improve the efficiency and effectiveness of OvS strategies.

Heterogeneous

Heterogeneous Simulation Optimization

Exploits the computational power of CPUs and GPUs to perform the distributed evaluation of simulations.

Homogeneous

Automatic Calibration of ABM Simulations

Exploits the computing power of a homogeneous cluster of nodes to calibrate an ABM simulation through MASON and ECJ.

Cloud

Cruise Initinerary Scheduling Design Optimization.

Exploits the power of an Amazon AWS MapReduce cluster to solve the Cruise Itinerary Scheduling Design Problem.

Table of Contents

Heterogeneous Simulation Optimization (HSO) [4]

- HSO is a framework for developing OvS processes in a heterogeneous computing system composed of CPU and GPU.
- HSO is an open-source project available on GitHub.
- HSO is developed for:
  - Complex simulations (running a single simulation takes longer than the optimization process).
  - The number of configurations to be evaluated is high.

[4] Cordasco, G.; D'Auria, M.; Spagnuolo, C. & Scarano, V., "Heterogeneous Scalable Multi-languages Optimization via Simulation", Communications in Computer and Information Science, Springer Verlag, 2018, 18th Asia Simulation Conference, AsiaSim 2018.

Heterogeneous Simulation Optimization

- Master / Worker architecture based on the Message Passing paradigm and developed in C.
- Support for:
  - Heterogeneous Hardware: Allowing to exploits the advantage of CPU and GPU.
  - Heterogeneous software: optimizer and simulator can be described in different programming languages.
  - Scalability: Allowing to leverage an increasing number of resources.

HSO Performance evaluation

- Heterogeneous Computing Cluster:
  - 14 CPU: 7 machines with 2 x Intel (R) Xeon (R) CPU E5-2680 2.70GHz and 256GB RAM.
  - 8 GPUs: 3 machines with 2 x Tesla M2090 and 2 machines with 1 x Tesla M2090.
- Examples:
  - Spread of Information: epidemiological, behavioral model in which the agents' behavior follows in the SIR model: susceptible, infected, recovered.
  - Spread of Influence: a social influence model in which individuals change their attitudes and behaviors in agreement with other individuals.

Spread of Information

- CPU simulation: Netlogo.
- GPU simulation: FlameGPU.
- Optimization process: Genetic Algorithm built through Python's DEAP library.
- Total evaluations carried out: 1284.

Spread of Influence

- CPU simulation: C.
- GPU simulation: Nvidia CUDA.
- Optimization process: Optimal Computing Budget Allocation written in Java.
- Total evaluations made: 200000.

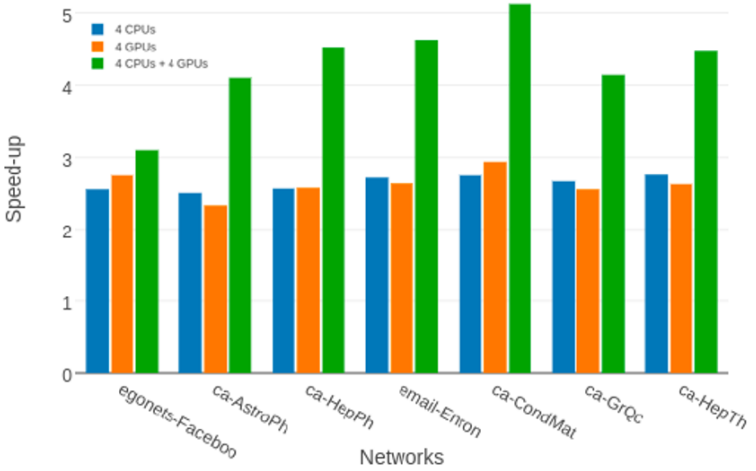


Table of Contents

The Agent-Based Model Calibration Problem

- A big challenge in ABM is the Model Validation.
- Choosing parameter values (model calibration) is a big part of validation.
- Despite its importance, the calibration is often done by hand using guesswork and manual tweaking, or the model is left uncalibrated.
- Problems:
  - Many parameters.
  - Intricate interactions.
  - Complex behaviors.
  - Slow to run.

Automated calibration process with MASON and ECJ [5] [6]

- Tools used:
  - MASON: library for the realization of ABM simulations;
  - ECJ: library of optimization metaheuristics.
- The calibration process:
  - Can be massively distributed.
  - Accesses a large number of optimization methods through ECJ.
  - Optimizes standard templates written in MASON without the need for modifications.

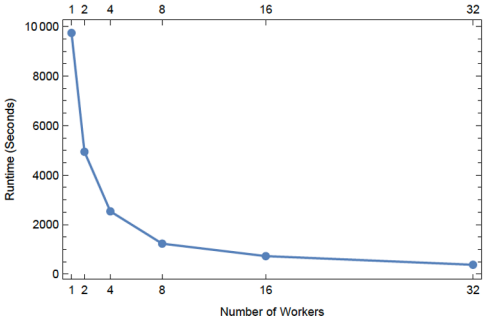
[5] D'Auria, M.; Scott, E.; Lather, R.; Hilty, J. & Luke, S., "Distributed, automated calibration of agent-based model parameters and agent behaviors", Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2020.

[6] D'Auria, M.; Scott, E.; Lather, R.; Hilty, J. & Luke, S., "Assisted Parameter and Behavior Calibration in Agent-Based Models with Distributed Optimization", Lecture Notes in Computer Science, Springer, 2020. 18th International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAMS 2020

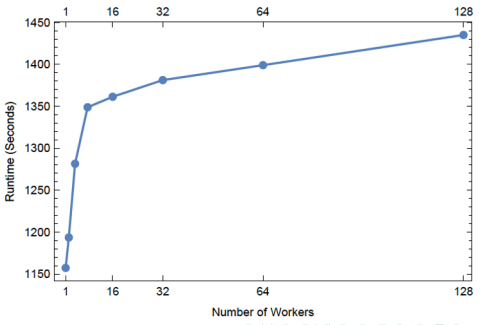
Scalability Analysis

- Homogeneous calculation system:
  - 24 machines, each with 2 x Intel Xeon E5-2670@2.60GHz and 24GB of RAM.
- Refugee: Explore the pattern of refugee migration in the Syrian crisis.

Strong Scalability Efficiency: 71,88%.



Weak Scalability Efficiency: 83,18%.



Optimization of Agent Behaviors

- Not just parameters can be optimized, but agent behaviors as well.
- The modeler is responsible for designing how behavior operates.

- Serengeti Model:
  - Gazelle: Obstacle-Avoidance behavior to evade lions.
  - Lions: Genetic Programming Parse-Tree which return a vector indicating speed and direction.
- The optimizer is a Genetic Programming algorithm with a population size of 5760 individuals.

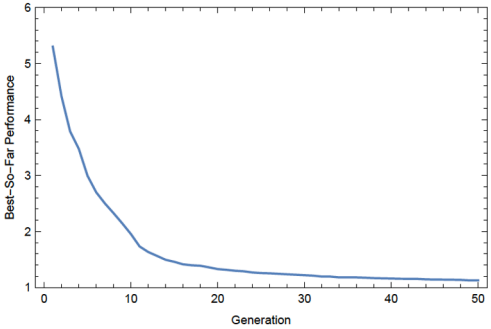
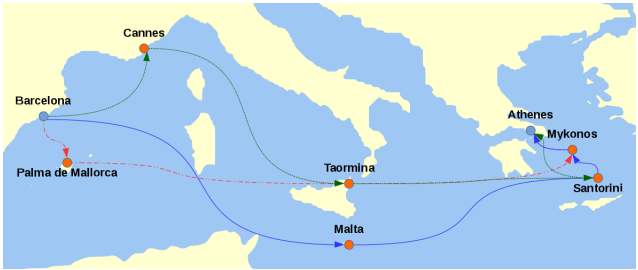


Table of Contents

Cruise Itinerary Scheduling Problem

- Multi-objective optimization process.
- Stages:
  - Ship allocation.
  - Cruise Scheduling.
  - Day-by-Day Optimization.



- Objective: minimizing the total cost C of the cruise and maximizing the commercial attractiveness A of the cruise.

Large-scale Optimized Searching for Cruise Itinerary Scheduling on the Cloud [7]

- A heuristic search algorithm was developed for the day-to-day optimization phase.
- The algorithm exploits:
  - a Tabu-Seach Algorithm to identify the cruise itinerary.
  - a Genetic Algorithm to optimize the search algorithm parameter.
- We used the SOF framework to deploy the OvS process on an Elastic MapReduce (Amazon AWS) cluster.

[7] Carillo, M.; D'Auria, M.; Serrapica, F.; Spagnuolo, C.; Caligaris, C. & Fabiano, M., "Large-scale Optimized Searching for Cruise Itinerary Scheduling on the Cloud", 2019 International Conference on Optimization and Applications, ICOA 2019, Institute of Electrical and Electronics Engineers Inc., 2019

Performance Evaluation

- Elastic MapReduce cluster with 34 EC2 nodes (c4.2xlarge):
  - 8 vCPUs, 15 GB of RAM, 1000 Mb / s of bandwidth.

- Speedup: x13.
- Cost: x2.4.

Nodes	vCPUs	Memory	time(s)	Cost on demand EC2	Cost on demand EMR	Total cost
1	4	15.0 GB	27165	$ 3.00	$ 0.79	$ 3.79
1	8	15.0 GB	23164	$ 2.56	$ 0.68	$ 3.24
2	16	30.0 GB	18695	$ 4.13	$ 1.09	$ 5.22
4	32	60.0 GB	10185	$ 4.50	$ 1.19	$ 5.69
8	64	120.0 GB	5259	$ 4.65	$ 1.23	$ 5.88
16	128	240.0 GB	3610	$ 6.39	$ 1.68	$ 8.07
32	256	480.0 GB	2005	$ 7.09	$ 1.87	$ 8.96

Table of Contents

Conclusion

- Computational Science is an attractive and active research field.
- Aims to solve complex problems.
- Solutions to these problems are usually expensive and time-consuming.
- The notion of scalability becomes central.
- From the point of view of computational scientists creating scalable solutions is not easy.

Conclusion

- The first difficulty that a computational scientist encounters may be related to the choice of the programming language.
- Many languages provide parallel constructs, and each has unique features that might be of interest.
- In many cases, it is necessary to combine several languages together.
- With this in mind, the FLY Language has been developed to enable the development of computational workflow, providing a single language addressing the most common computational scientists' needs.

Conclusion

- Given the importance of ABMs, in the context of Computational Science, powerful simulation libraries are essential.
  - Rust-AB exploits the peculiarities of the Rust Language to create simulations with a very high density of agents.
- Finally, but no less critical, the necessity to have frameworks for OvS processes exploiting the computing system available to the computational scientists:
  - Heterogeneous, where simulations are executed in a distributed system composed of heterogeneous nodes.
  - Homogeneous, where MASON and ECJ are used to elaborate the OvS process.
  - Cloud computing, where a MapReduce cluster runs an OvS process.
- All presented solutions are available on GitHub under an Open-Source license.

Thanks for the attention!!!