# New Trends in Parallel and Distributed Simulation:

*Many-cores, Cloud Computing and Energy Efficient Simulation*

**Gabriele D'Angelo**

*g.dangelo@unibo.it*

*http://www.cs.unibo.it/gdangelo/*

*Department of Computer Science and Engineering*

*University of Bologna, Italy*

***joint work with M. Marzolla***

**2016 @ National University of Singapore**

# Seminar **slides**

- These slides can be found at the following URL

http://nus.portazero.it

# Seminar **outline**

- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

# Seminar **outline**

- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

# Starting from scratch: **simulation**

- **"A computer simulation is a computation that models the behavior of some real or imagined system over time"**
  **(R.M. Fujimoto)**
- **Motivations:**
  - performance evaluation
  - study of new solutions
  - creation of virtual worlds such as online games and digital virtual environments
  - ...

# **D**iscrete **E**vent **S**imulation (**DES**)

- The **state** of the simulated system is represented through a **set of variables**
- The key concept is the "**event**"
- An event is a **change in the system state** and it **occurs at an instant in time**
- All is done through the **creation**, **delivery** and **computation** of events
- The computation of an event can **modify some part of the state** and **lead to the creation of new events**

# **DES** on a **single CPU**: **sequential simulation**

- All simulation tasks are accomplished by a **single execution unit** (that is a CPU and some RAM)
- **PROS**: it is a **very simple approach**
- **CONS**: there are a few **significant limitations**
  - the **time** required to complete the simulation run
    - *how fast is a single CPU?*
    - *in some cases results have to be in real time or even faster!*
  - if the model is quite large and detailed the RAM is not sufficient: it is **not possible to model some systems**
- **This approach does not scale!**

# Going **Parallel**: **P**arallel **D**iscrete **E**vent **S**imulation

- **Multiple interconnected** execution units (**CPU**s or **hosts**)
- Each unit manages **a part of the simulation model**
- Very large and complex models can be represented using the resources **aggregated** from many execution units
- **Locally generated events** may have to be **delivered to remote execution units**
- All of this needs to be carefully **synchronized**
- "**Concurrent events**" can be **executed in parallel**, this can lead to a significant **speedup of the execution**

# Seminar **outline**

- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

# **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)

- There is **no global state**: this is the key aspect of **PADS**
- A **PADS** is the **interconnection** of a set of **model components**, usually called **Logical Processes** (**LP**s)
- Each **LP** is responsible to manage the evolution of only **a part of the simulation**
- Each **LP** has to interact with other **LP**s for **synchronization** and **data distribution**
- In practice, each **LP** is usually executed by a **processor** (or a **core** in modern multi-core architectures)
- The **cost of communication** among **LP** can be very high
- There can be fault-tolerance problems

# On the (lack of) **global state** and its **consequences**

- The model has to be **partitioned** in **components** (the **LP**s)
- In a parallel/distributed architecture **synchronization** mechanisms have to be implemented
- **Data is produced locally** (within the **LP**) but can be of interest to other parts of the simulator (other **LP**s): **data distribution** mechanisms
- All these are **main problems of PADS**: we need to introduce them more in detail

# **Partitioning**: creating and allocating parts

- Each **LP** is responsible for the management of **a part of the simulated model**
- In some cases the partitioning follows the **structure** and the **semantics** of the simulated system
- In other cases is **much harder**, for example if the system is **monolithic** and hard to split in parts
- **Many different aspects** have to be considered in the partitioning process
- For example:
  - **minimization** of **network communication**
  - **load balancing** of both **computation** and **communication** in the execution architecture

# **Synchronization**: on the correct order of events

- Some kind of **network** interconnects the **LP**s running the simulation
- Each **LP** is executed by a different **CPU** (or **core**), **possibly at a different speed**
- The **network** can **introduce delays** but we assume that the communication is reliable (e.g. TCP-based communications)
- The results of a **PADS** are **correct** only if its outcome is **identical** to the one obtained from the corresponding **sequential** simulation
- **Synchronization mechanisms** are used to **coordinate** the **LP**s: **different approaches are possible**
- This task usually has a **very relevant cost**

# **Data distribution**: on the dissemination of information

- Each component of the simulator will produce **state updates** that are possibly **relevant for other components**
- The distribution of such updates in the execution architecture is called **data distribution**
- For overhead reasons **broadcast can not be used**
- The goal is to **match** data **production** and **consuming** based on **interest criteria**
- Only the **necessary** data has to be **delivered** to the **interested** components
- There are both **communication** and **computation** aspects to consider
- Data distribution also has to be properly synchronized

# Seminar **outline**

- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

# **New challenges**: what's next?

- Evolution in computing technology is fast and often confusing
- But it is possible to identify some **characteristics** and **trends**
- Frequent **updates in hardware** but **software is slow in supporting them**
- On the other hand, **software** is **limited** by **hardware characteristics**
- For many years, 32 bits processors have limited the max amount of memory of sequential simulators
- Now with 64 bits CPUs memory remains an issue only with huge simulations

## **New challenges**: some existing and new trends

- The so called "**MHz race**" in CPUs has **slowed down**
- **Multi-core CPUs** are now available at bargain prices
- Only few users have access to **High Performance Computing** facilities *(i.e. supercomputers and dedicated clusters)*
- Many are willing to use **Commercial Off-The-Shelf** (**COTS**) hardware that is also **shared with other tasks** *(e.g. desktop PCs or underloaded servers)*
- **Outsourcing** the **execution of simulations** is the next big step in this direction *(e.g. simulation as a service)*

# **New challenges**: cloud computing

- **Cloud computing** is a model for providing **on-demand** network access to a **shared pool** of computing resources
- Such resources can be **provisioned** and **released quickly** and with minimal management effort
- For many reasons cloud computing is becoming mainstream
- Implements the "**pay-as-you-go**" approach: virtual computing environments in which you **pay only for capacity that you actually use**
- The resources are obtained from a shared pool and provided by **commercial service providers**

# **New challenges** vs. **existing software tools**

- **(Most of) available simulators** are **unable to cope** with such changes in the execution environment
- Often they **do not exploit** all the **available resources**
- That means that are **too slow** in obtaining the **results**
- The effect is that users are more and more encouraged to **oversimplify the simulation models**
- That's a very **risky move**...

# **New challenges**: many cores

- **Entry level CPUs** provide ***2*** or ***4*** **cores** but processors with >***16*** cores are available on the market
- **Example:** EZchip TILE-Mx: **100** core ARM Cortex-A53
- This is a **big change in the execution architecture** and will **not be transparent** to simulation users
- **Sequential simulators** are, for the most part, **unable to exploit more than one core** *("offloading" is **not** the best solution)*
- This means that **PADS** techniques will be **necessary** even to run simulations on a **desktop PC**

# **New challenges**: many cores

- Even if assuming that all **cores** are **homogeneous** *(and that is not always true)*, the simulation model has to be **partitioned** in **more and more LP**s
- The **partitioning** is a **complex task** and increasing the number of cores it becomes **harder** and **harder**
- The **load of each core** has to be **balanced** and the **communication** among cores has to be **minimized**
- Who is in charge of the partitioning has to predict *a priori:*
  - *the **behavior** of the **simulated model***
  - *the **load** of the **execution architecture***

# **New challenges**: many cores

- All **static approaches** are **suboptimal**: the **runtime conditions** are **variable**
- Who is in charge of **partitioning**?
  - *currently, the **software** is **unsuitable** to perform this task*
  - *it is still in charge of the **simulator user**!*
- It is clear that **this approach does not scale**!
- Most simulation users are not willing to become experts of **PADS** or computing architectures
- Their goal is to **obtain results as fast as possible** and **with the least effort**
- It is clear that it should be a **software task**!

# **New challenges**: the public cloud

- Everything is going "**on the cloud**". Why simulation is **not**?
- Please do not confuse the **private cloud** and the **public cloud** infrastructures, they are very different!
- The big goal is to follow the "**everything as a service**" paradigm and to **rent the resources** for running simulations
- On the market there are many providers of cloud services *(e.g. Google, Amazon, Microsoft...)*
- You **pay only for the rented resources** and you can **increase** or **decrease** them **dynamically**
- This is great for small or medium size firms: **no more investments in hardware**!

# **New challenges**: the public cloud

- A public cloud environment can be **very dynamic**, **variable** and **heterogeneous**
- The **virtual instances** providing the services can be located in **different data centers**, with **different Service Level Agreements** and from **different providers**
- Under the **PADS** viewpoint, **it is again a matter of partitioning**
- Are current synchronization protocols *(e.g. conservative vs. optimistic)* able to deal with a cloud environment?
- Again… the **software tools** that are available on the market are **unable** to cope with this problem

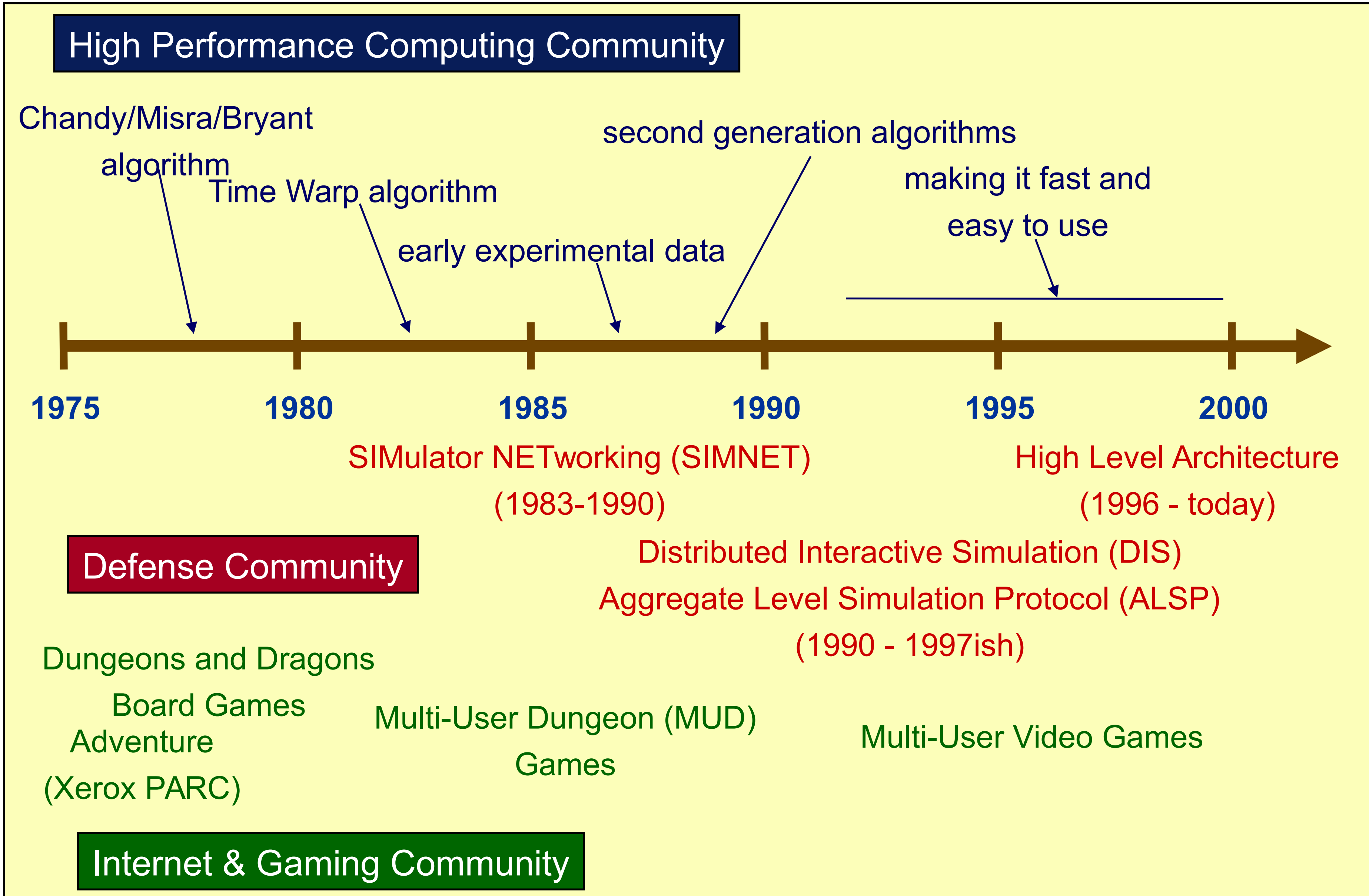
# **New challenges**: the public cloud on steroids

- Let's go on with our vision of "**simulation-as-a-service**"
- The price of cloud computing services is highly dependent on aspects such as **reliability** and **guaranteed performance**
- It is a pricing model based on the assumption that all customers have the **same requirements**
- **PADS** tools could (automatically) rent very **inexpensive** (and **low reliability**) cloud services
- The **middleware** running the **PADS** will be in charge of **coping with faults**
- This can be "easily" done adding some degree of **replication**
- This is a further extension of the partitioning problem

# Seminar **outline**

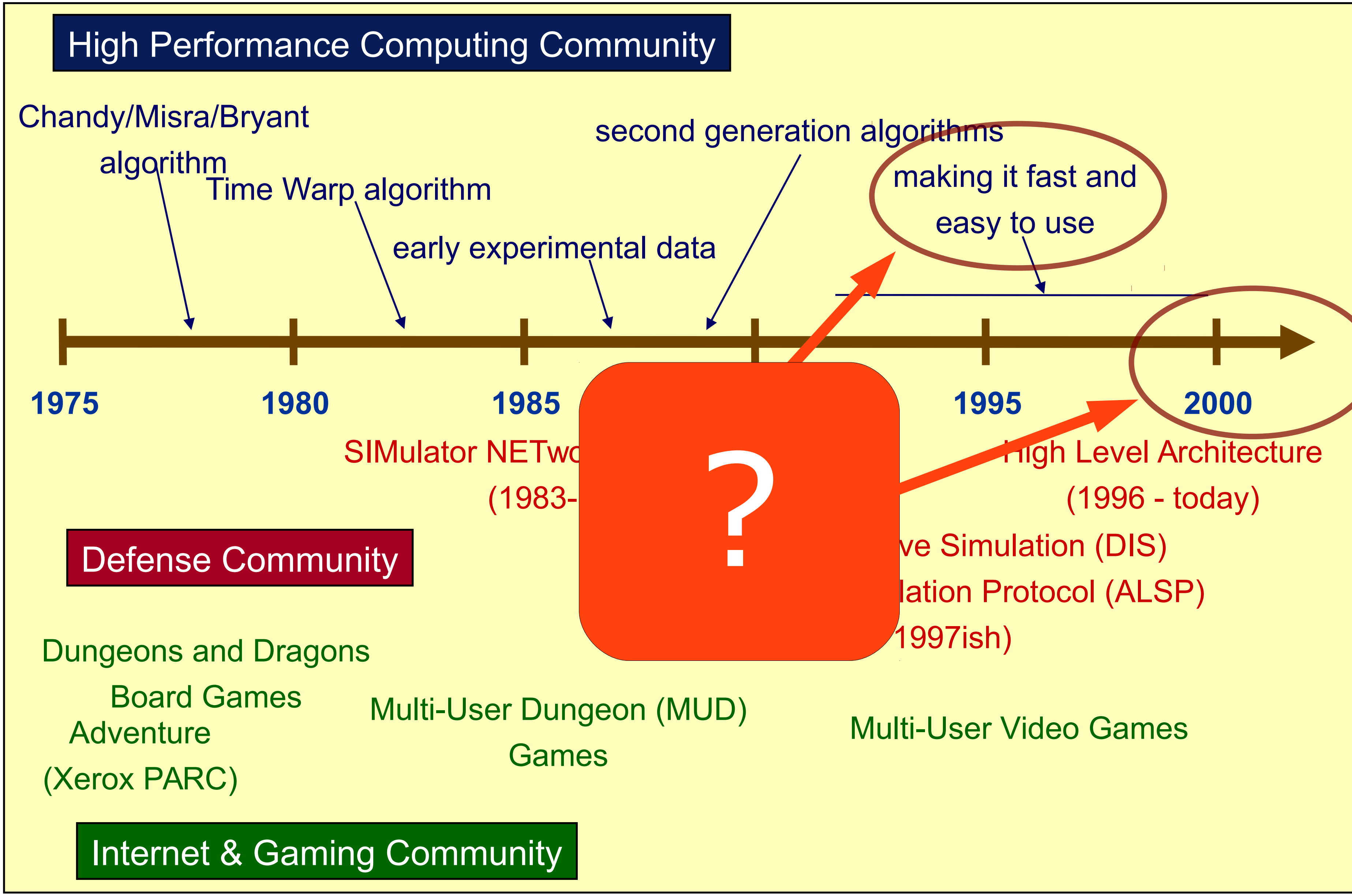
- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

# Historical perspective on **PADS**




Richard M. Fujimoto, tutorial, 2000

# Historical perspective on **PADS**




Richard M. Fujimoto, tutorial, 2000

# **PADS**: what happened in the last two decades?

- Two **main research goals**:
  - *make it **fast***
  - *make it **easy** to use*
- A lot of work in synchronization and data dissemination management has been done

  → **in some conditions** PADS is very fast

  *… properly partitioned model, appropriate synchronization algorithm, homogeneous execution architecture …*
- What about **usability**?

  **PADS does not work straight out of the box**
- The level of **knowledge** modelers are required is still too high, some aspects are **hard to manage** and **understand**

# **PADS**: cost assessments, the need for new metrics

- The **amount of time** needed for completing a simulation run is called **W**all-**C**lock-**T**ime (**WCT**)
- The **WCT** has always been the **main metric** to evaluate the **efficiency of simulators**
- This can be right in classic execution architectures but **it is not** when the resources are obtained following the "***pay for what you use***" scheme *(e.g. public cloud)*
- A **more complex evaluation** has to be done:
  - *how much time the user can **wait for the results**?*
  - *how much he **wants to pay** for running the simulation?*
- Are the current **PADS** algorithms and mechanisms suitable for this new evaluation metric?

# **PADS**: in search of performance

- Let's assume that **costs are not a problem** and that the goal is to **obtain the results as fast as possible**
- Continuing to focus on **synchronization**, the traditional algorithms are **fast** when run in a **public cloud**?
- What level of **performance** should we expect?
- The answer is quite simple: using the traditional approaches the **obtained results are often (very) poor**
- What is the **problem**?

# Seminar **outline**

- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

# **How**: on the **adaptive approaches**

- **Warning**: the "silver bullet" does not exist, even in simulation
- In our vision, all starts with the **partitioning problem**: *decomposing the simulation model into a number of components and properly allocating them among the execution units*
- **Constraints**: the **computation load** has to be kept **balanced** while the **communication overhead** has to be **minimized**
- Given that the runtime conditions are largely **unpredictable** and the environment is **dynamic** and very **heterogeneous**, all **static approaches are not adequate**

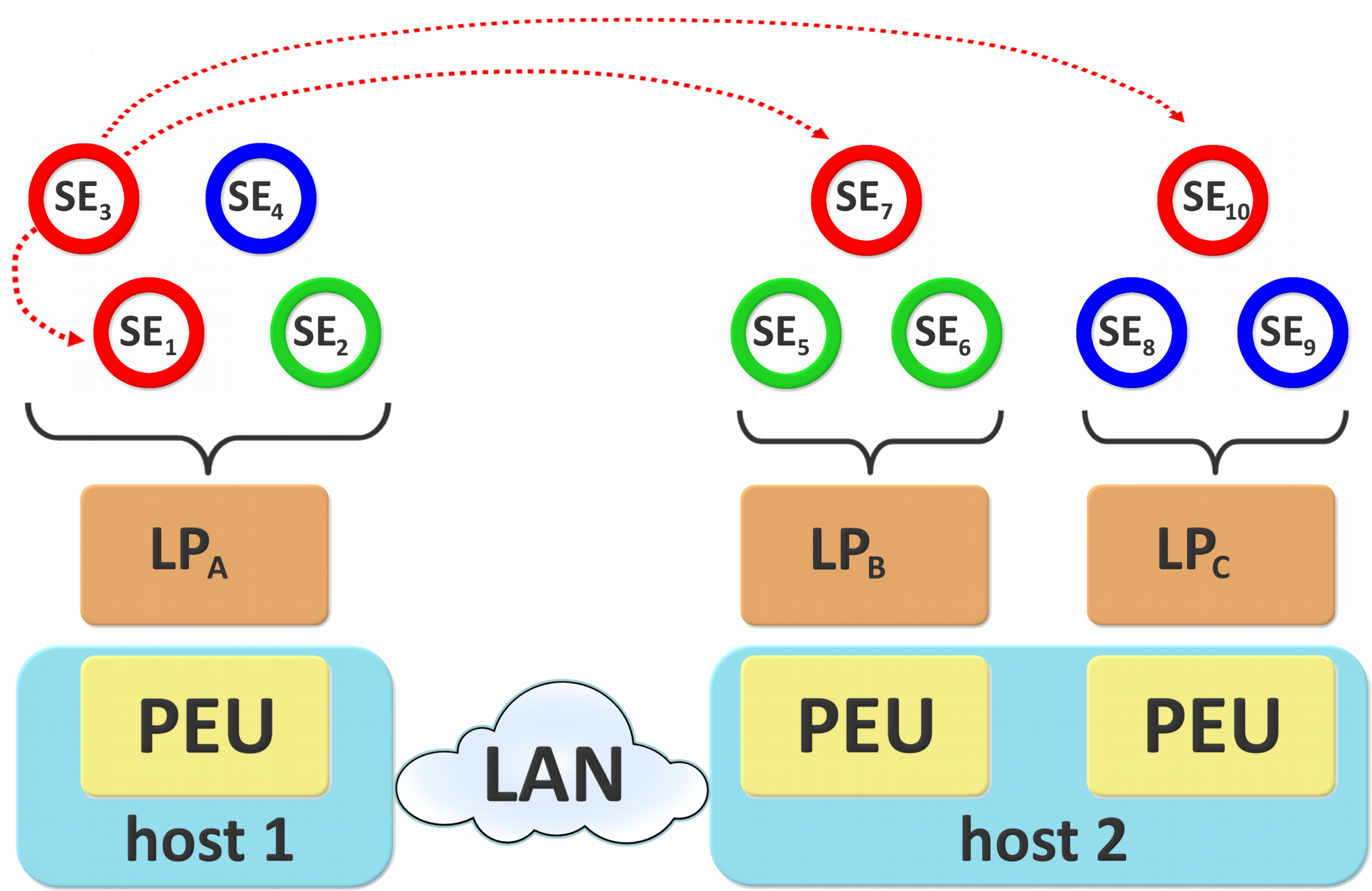
# **Migration**-based **adaptive partitioning**

- The simulated model is divided into **very small parts** (called **Simulated Entities**, **SE**s)
- Each **SE** is a tiny piece of the simulated model and interacts with other **SE**s to implement the model behavior
- It is some sort of **Multi Agent System** (**MAS**)
- Each **node** (called **Logical Process, LP**) in the execution architecture is the container of a dynamic set of **SE**s
- The **SE**s are **not statically allocated** on a specific **LP**, they can be **migrated** to:
    - *reduce the **communication overhead***
    - *enhance the **load balancing***

# Adaptive clustering: **migration of entities**

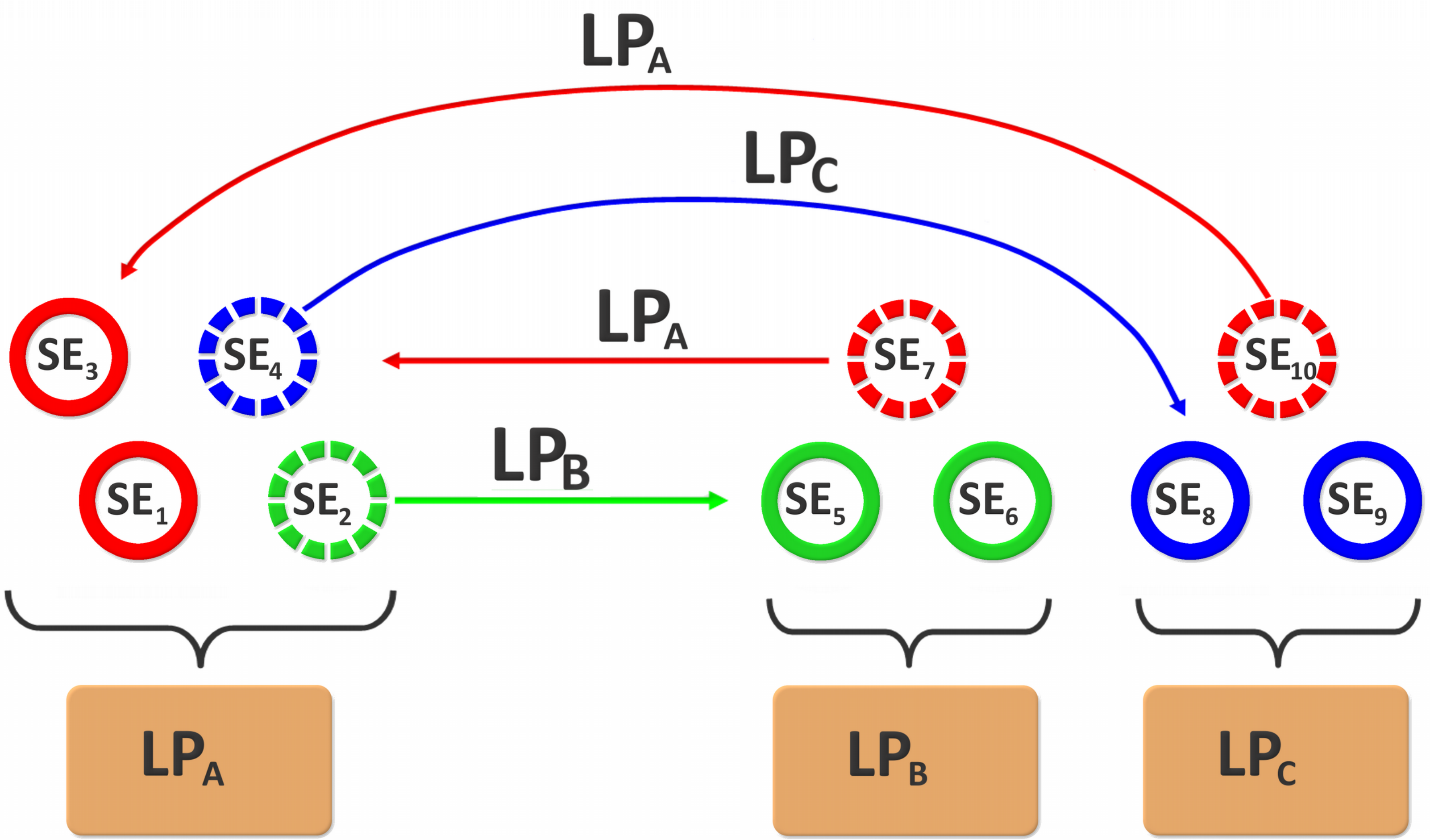
- In a parallel/distributed simulation the **communication overhead** is usually quite high
- Each **SE** will have (possibly) different **interaction patterns**
- In the simulation, it is possible to find "**interaction sets**" composed of **SE**s **interacting with high frequency**
- The main strategy is to **cluster** the **SE**s interacting with high frequency **within the same LP**
- All of this can be done **analyzing the communication pattern** of each **SE** and **migrating some of them**
- **The load balancing has to be considered!**

# Adaptive clustering: **migration of entities**



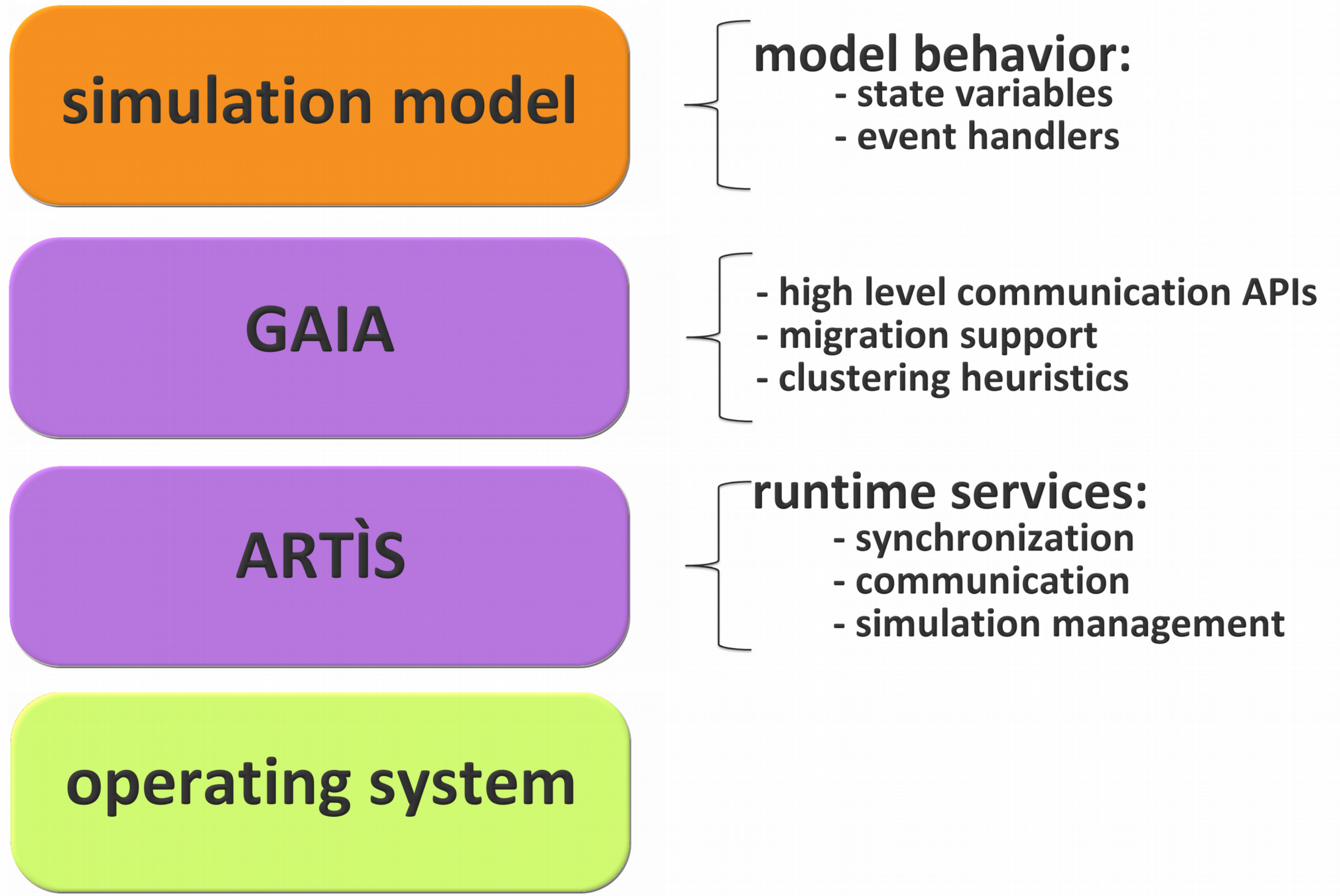

In dashed lines, the **interactions** of $SE_3$ with other simulated entities

# Adaptive clustering: **migration of entities**




In solid lines, the **migrations** that should be done to enhance the partitioning

# The ARTÌS/GAIA simulator



# **ARTÌS** and **GAIA**, some details

- Multi-year effort for building an **efficient simulation middleware**: **A**dvanced **RTI S**ystem (**ARTÌS**)
- Used as a testbed for many research works
- The **G**eneric **A**daptive **I**nteraction **A**rchitecture (**GAIA**) framework implements the **adaptive features**:
  - ***adaptive clustering*** *for* ***overhead reduction***
  - ***dynamic load balancing*** *of communication and computation*
  - *support for* ***heterogeneous*** *execution platforms and* ***shared computing resources***
  - ***Reliable-GAIA****: support for* ***fault-tolerance*** *(work in progress)*

**For details** and **software download**: http://pads.cs.unibo.it

# **ARTÌS** and **GAIA**, some details

- Multi-year effort for building an **efficient simulation middleware** ... (**ARTÌS**)
- Used as a testbed for many research works
- The **G**eneric **A**daptive **I**nteraction **A**rchitecture (**GAIA**) framework implements the **adaptive features**:
  - ***adaptive clustering*** *for* ***overhead reduction***
  - ***dynamic load balancing*** *of communication and computation*
  - *support for* ***heterogeneous*** *execution platforms and* ***shared computing resources***
  - ***Reliable-GAIA****: support for* ***fault-tolerance*** *(work in progress)*

It is **free** for **education** and **research** purpose!

Many parts are provided with **source code** (e.g. all the simulation models)

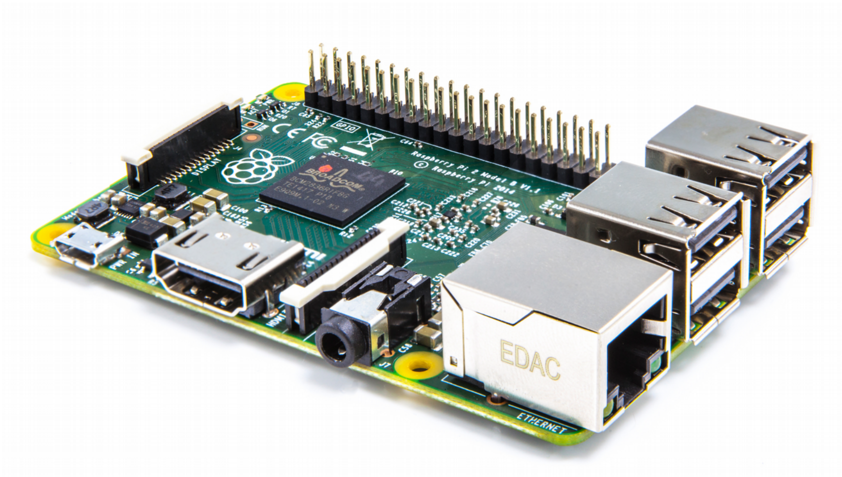
Our goal is to **Open Source** as soon as possible the whole software stack

**For details** and **software download**: http://pads.cs.unibo.it

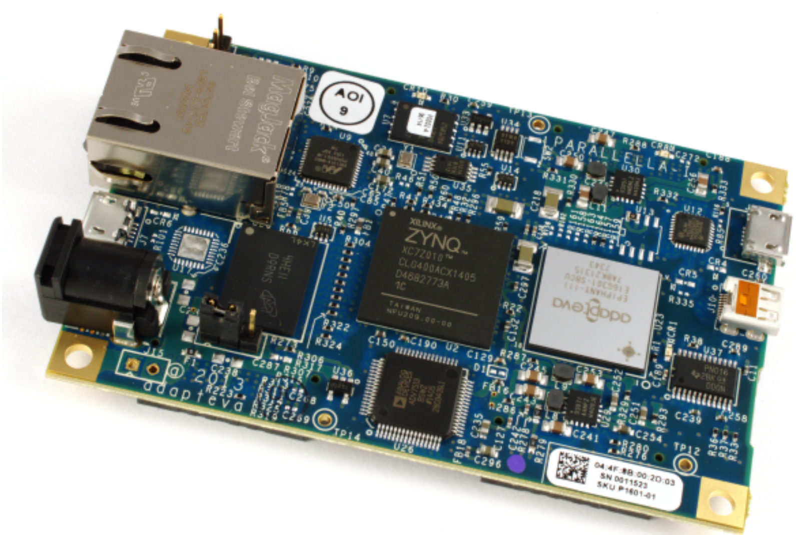
# Seminar **outline**

- **Background**
- **P**arallel **A**nd **D**istributed **S**imulation (**PADS**)
- New **challenges** of today and tomorrow
- **Functionality** and **limitations** of current PADS approaches
- In the search of **adaptivity**: the ARTÌS/GAIA approach
- **Energy efficient** simulation

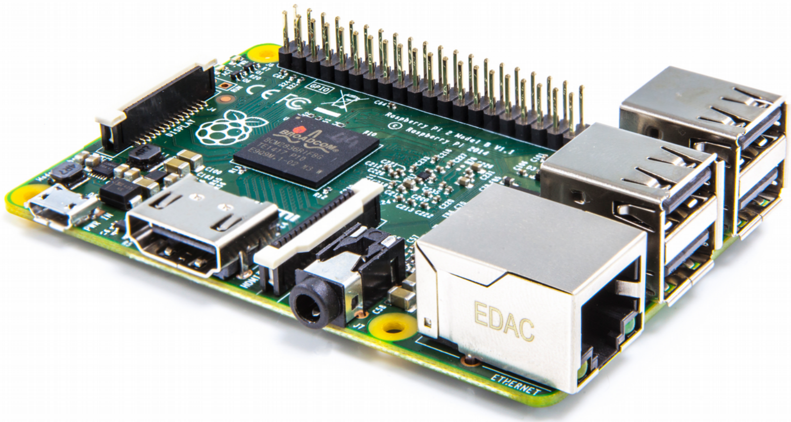
# Simulation **hardware**: **what is better?**

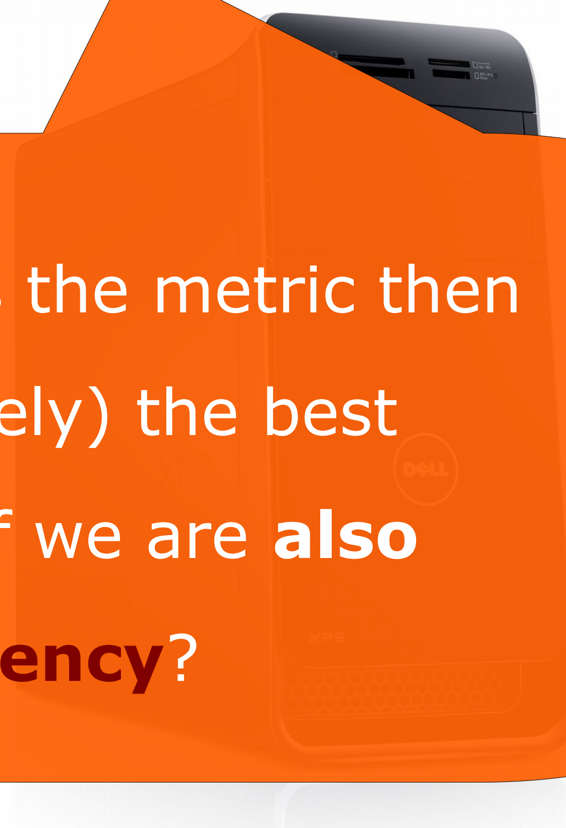






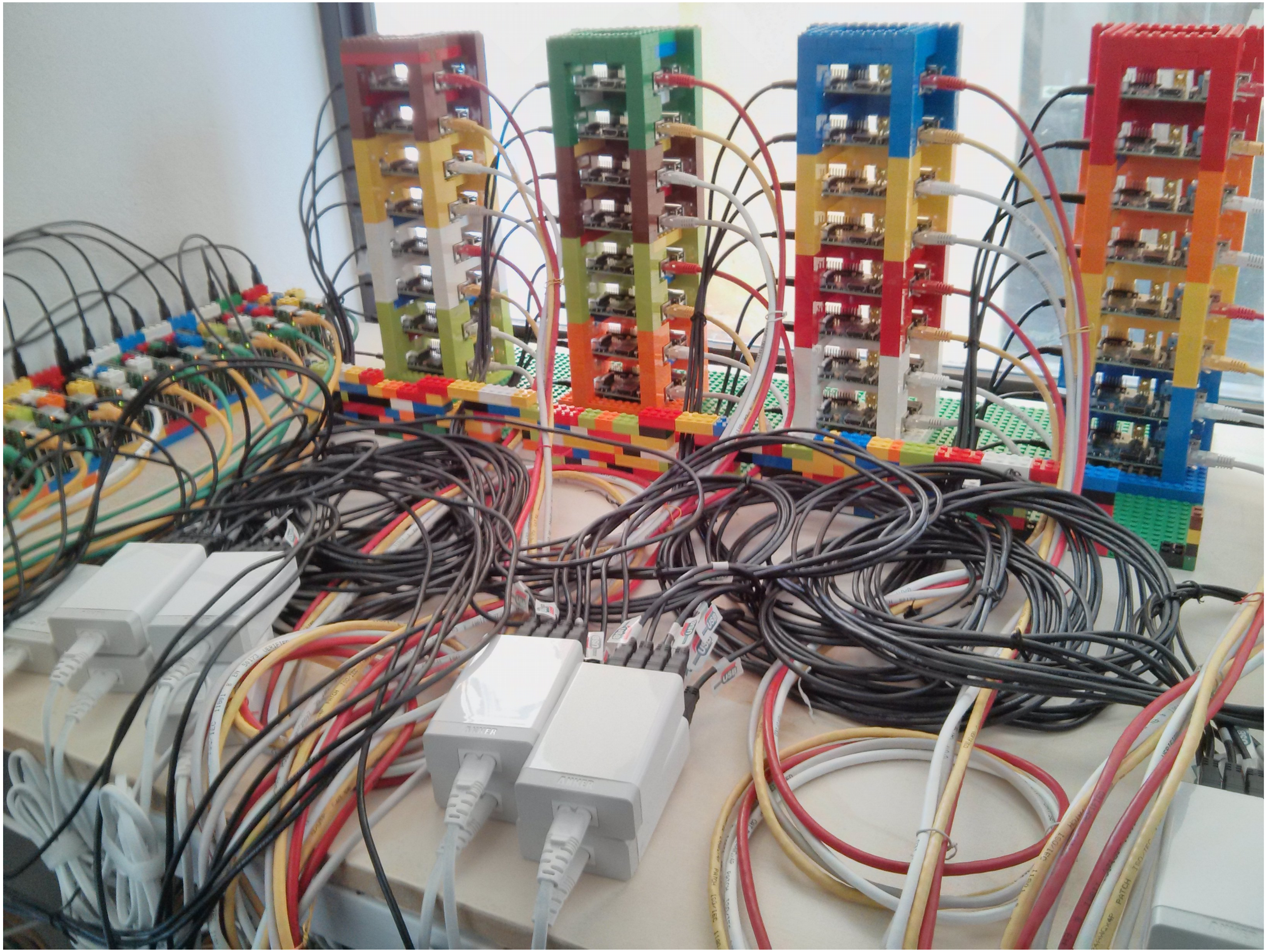

# Simulation **hardware**: **what is better?**



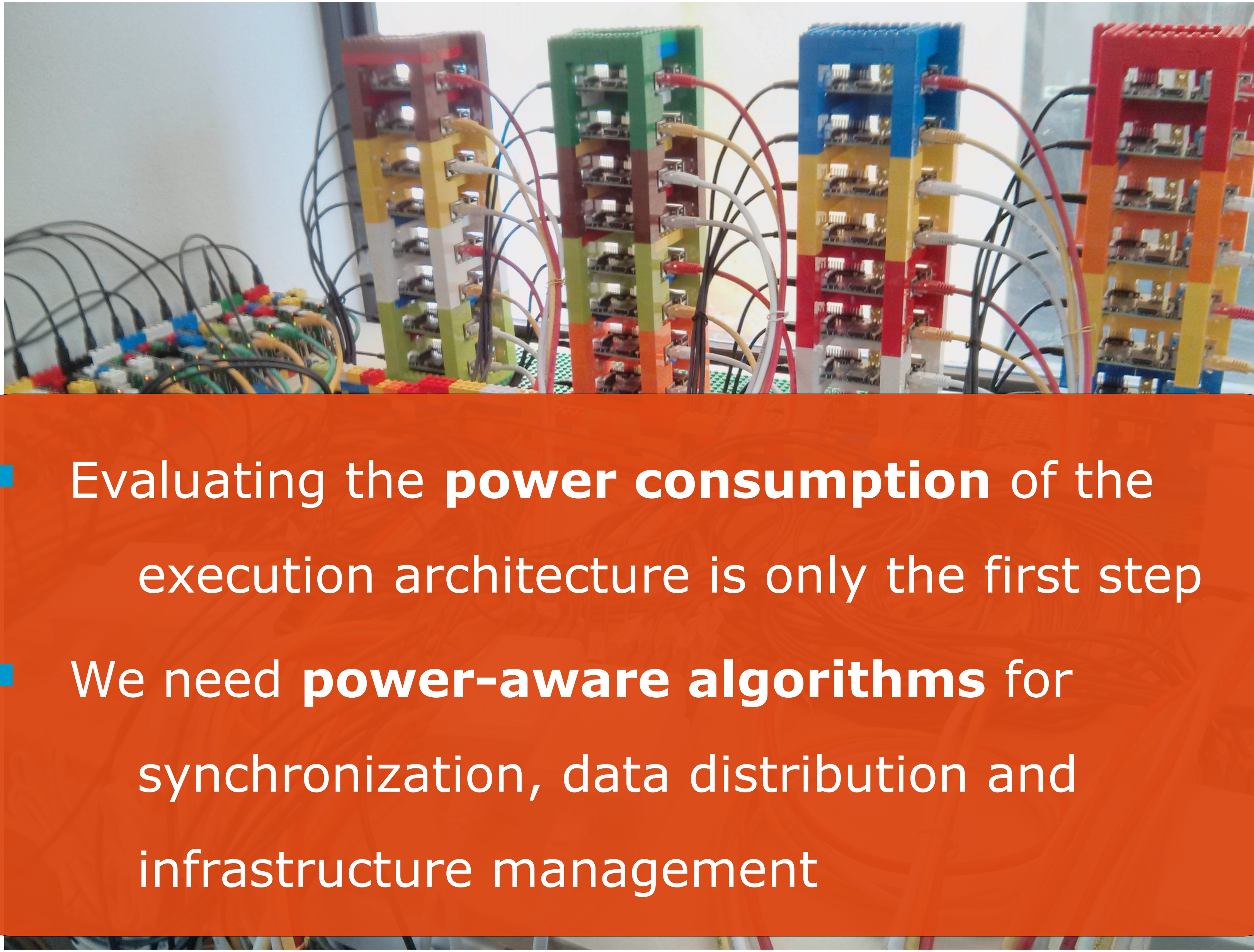



If the **execution speed** is the metric then the rack server is (very likely) the best choice but what happens if we are **also** interested in **power efficiency**?

# **Rapèin project**: http://raspein.cs.unibo.it



# **Rapèin project**: http://raspein.cs.unibo.it



- Evaluating the **power consumption** of the execution architecture is only the first step
- We need **power-aware algorithms** for synchronization, data distribution and infrastructure management

# Further information

Gabriele D'Angelo, Moreno Marzolla

**New Trends in Parallel and Distributed Simulation: from Many-cores to Cloud Computing (url)**

*Simulation Modelling Practice and Theory, Elsevier, vol. 49*

Gabriele D'Angelo

**The Simulation Model Partitioning Problem: an Adaptive Solution Based on Self-Clustering (url)**

*Simulation Modelling Practice and Theory, Elsevier, vol. 70*

The **ARTÌS** middleware and the **GAIA** framework can be downloaded from:

- http://pads.cs.unibo.it

**Gabriele D'Angelo**

- E-mail: <g.dangelo@unibo.it>
- http://www.cs.unibo.it/gdangelo/

# New Trends in Parallel and Distributed Simulation:

*Many-cores, Cloud Computing and Energy Efficient Simulation*

**Gabriele D'Angelo**

*g.dangelo@unibo.it*

*http://www.cs.unibo.it/gdangelo/*

*Department of Computer Science and Engineering*

*University of Bologna, Italy*

***joint work with M. Marzolla***

**2016 @ National University of Singapore**