EFFICIENT LOAD BALANCING MECHANISM FOR PARALLEL NON-SPATIAL ABMS

Cristina Peralta[1], Eduardo César[1], Andreu Moreno[1,2], Anna Sikora[1]

1 Universitat Autònoma de Barcelona, Departament d'Arquitectura de Computadors i Sistemes Operatius, Bellaterra, Spain
2 Escola Universitària Salesiana de Sarrià, Barcelona, Spain

Introduction

- Modern **distributed non-spatial ABMS frameworks lack built-in load-balancing** that minimizes communication as the simulation graph evolves over time
- Multilevel graph partitioners (**MGP**) like Zoltan, ParMETIS, and ParHIP provide **good-quality partitioning** that **minimizes communication** and **balances workload**. However, they require **full graph repartitioning during simulation** time when rebalancing.
- To rebalance using MGP, the **simulation halts** while the distributed MGP runs. Once the **new partition** is **obtained**, the **graph is rebuilt before resuming**.

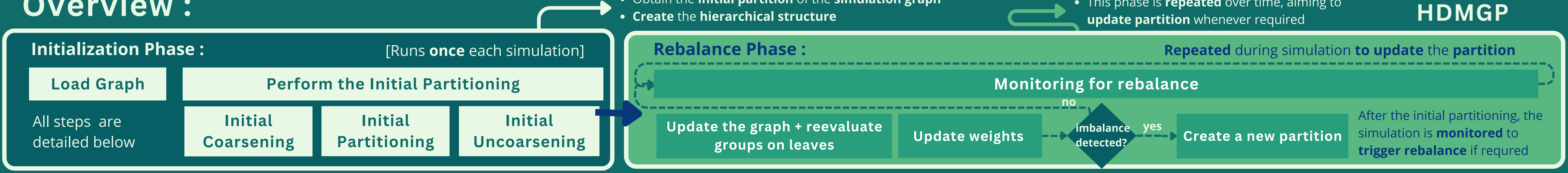
Proposal

- We propose building a hierarchical dynamic multilevel graph partitioning (**HDMGP**) tool to **balance non-spatial ABMS**
- Our tool aims to keep **multiple abstractions** of the graph at each level of the hierarchy and update **portions** of the graph when required
- This tool could be **run in a separate set of** processing elements **(PEs)** from the simulation ones, so the **monitoring and rebalance** is performed dynamically, **without halting** the simulation

HIERARCHICAL DYNAMIC MULTILEVEL GRAPH PARTITIONING

GOAL: *obtain an initial partition of the graph and update it as the simulation progresses*

Overview :



- Obtain the **initial partition** of the **simulation graph**
- **Create** the **hierarchical structure**

- This phase is **repeated** over time, aiming to **update partition** whenever required

HDMGP

Initialization Phase : [Runs **once** each simulation]

Load Graph

Perform the Initial Partitioning

All steps are detailed below

Initial Coarsening | **Initial Partitioning** | **Initial Uncoarsening**

Rebalance Phase : **Repeated** during simulation **to update** the **partition**

Monitoring for rebalance

Update the graph + reevaluate groups on leaves | **Update weights** | **Create a new partition**

After the initial partitioning, the simulation is **monitored** to **trigger rebalance** if requred

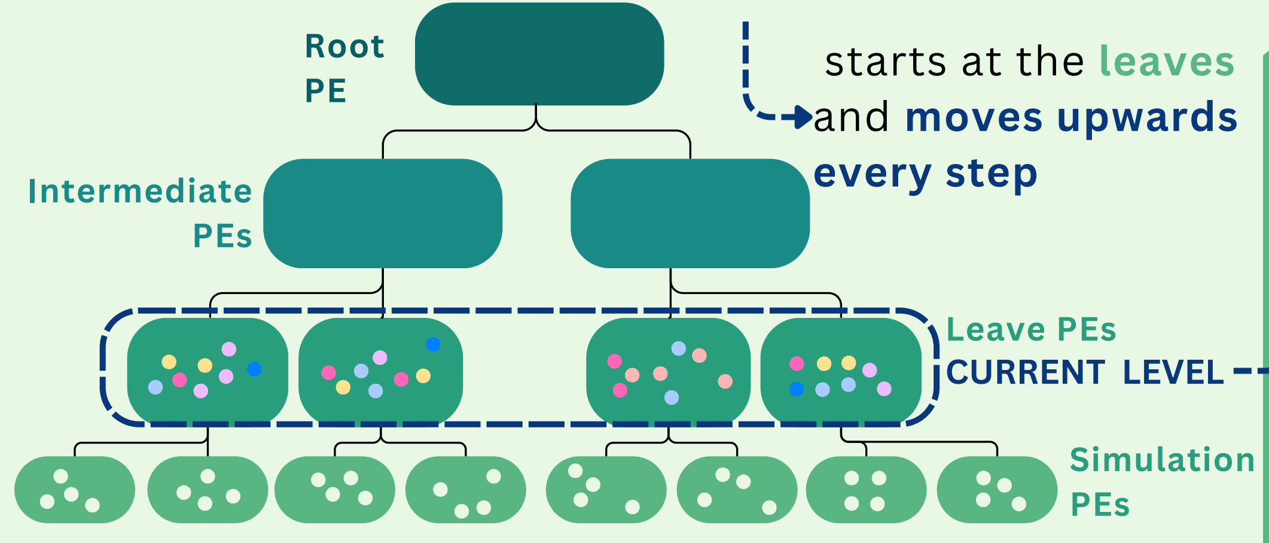
Initialization Phase

Initial Coarsening Phase

Initial Coarsening is executed once at the beginning of the simulation

1 Find Groups of Vertices

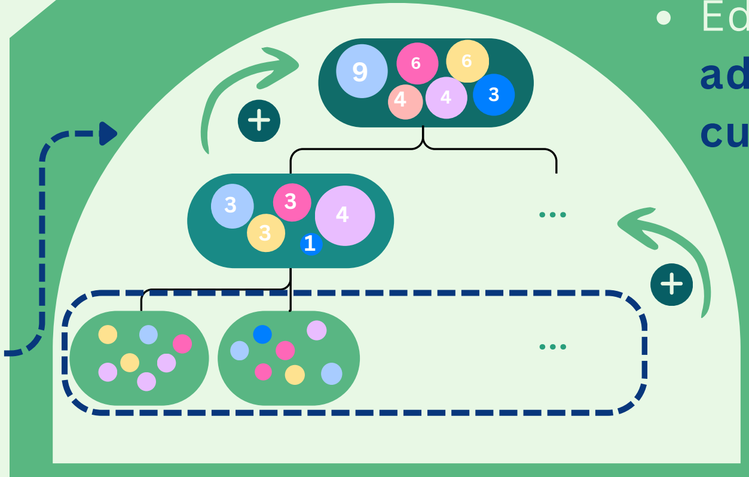
Group vertices in the **current level** using **Label Propagation Algorithm (LPA)**



starts at the **leaves** and **moves upwards every step**

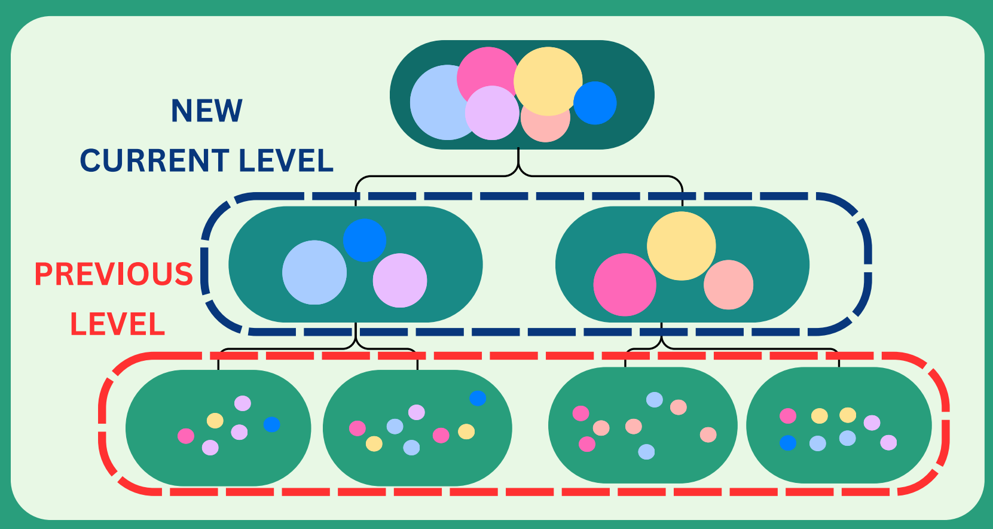
2 Agreggate Weights

- After finding the groups on the **current level** :
  - Edge & vertex **weights** are **added up** in each PE at the **current level**
    - Once added, weights are **forwarded** to the **next upper level**
  - At the end, the **root PE** held the **total weights** and the **count** of the **new vertices**



3 Create next level graph

As new vertices are found (one per group), a **new graph** is created at the **new current level**. This graph contains the new vertices and its edges (weighted)



- A **new current level** is set as the **parent** of the **previous one**
- **Found vertices** and their edges are **sent to the PEs** of the **new current level**
- The **new** distributed **graph is recreated** on **new current level**

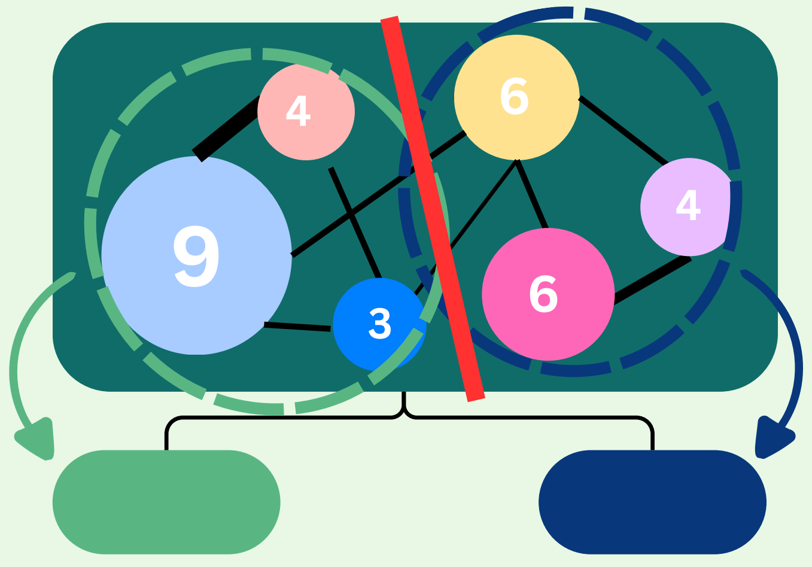
Not yet at the root

The root is reached

Initial Coarsening **ends** when the **root** owns a graph

Initial Partitioning Phase

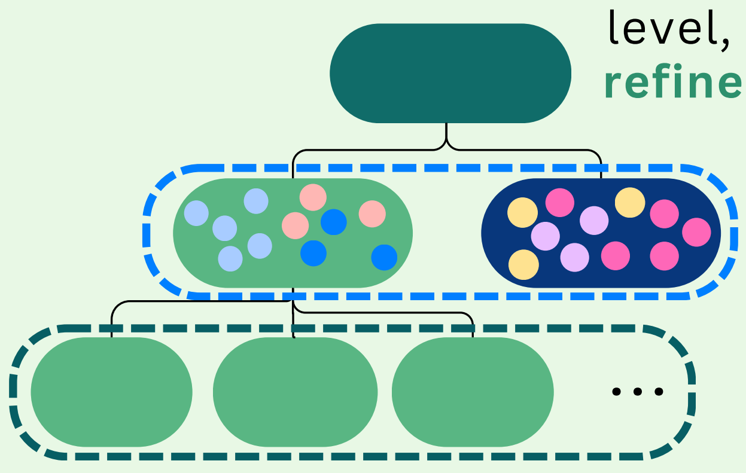
After Initial Coarsening, the partitioning begins at the root



- The **root's graph** is **divided** into **as many groups as it has children**
- **A group** is **assigned** to **each child**
- Every **child owns** the **vertices** in their **abstraction** of the graph that **belong** to its **assigned partition**

Initial Uncoarsening Phase

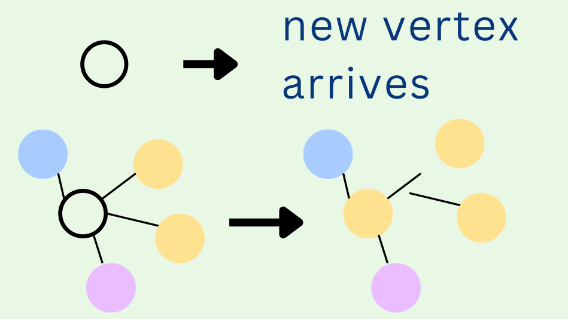
- The uncoarsening is an iterative process in which each level of the hierarchy **receives a partition** from its parent level, **projects** it into its vertices, **refines** it, creates a new partition for its children and forwards it to them
- This phase ends when the **leaves** of the graph **receive** and **refine** the **final partition** of the graph



Rebalancing Phase

Update the graph + reevaluate the groups on the leaves

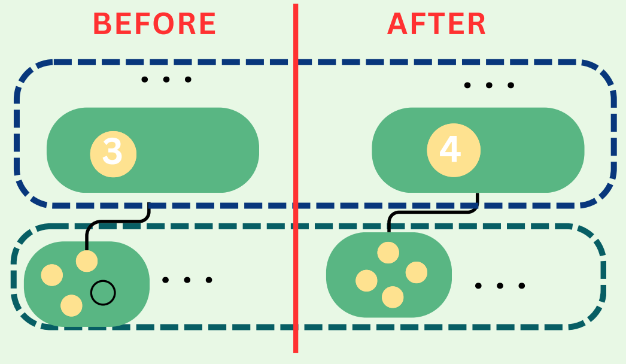
Once the hierarchy is created, the simulation can begin with the designated partition. As the simulation progresses, **the graph may change, requiring updates** to the partition **to maintain balance**.



Any **new vertex** arriving should be **assigned a group**, the **groups** of **vertices** whose **neighborhood change** must be **reevaluated**

Update Weights

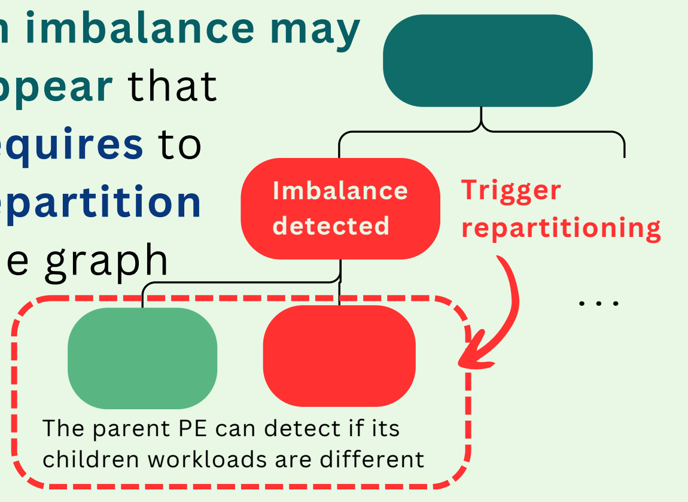
Assigning new vertices to a group **modifies its weight**



After reassessing groups on a graph change, **weight contributions must be recalculated** and **sent to parent levels** to update the total weight at all levels

Detecting Imbalances & Creating a new partition

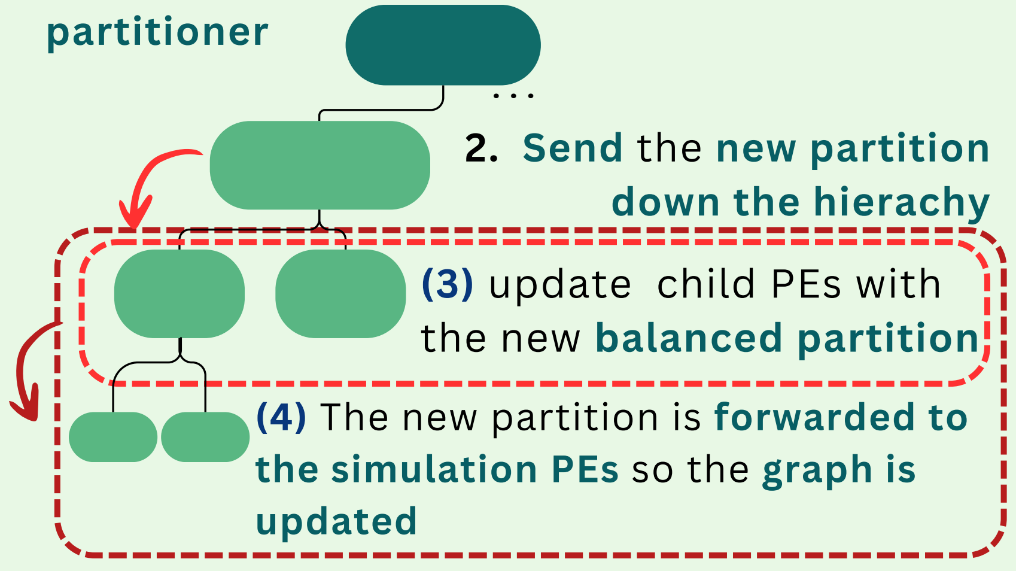
When changes on the graph occur, **an imbalance may appear** that **requires** to **repartition** the graph



PEs at higher levels **monitor for imbalances** among their children and **trigger repartitioning**.

Produce a **new partition** of the graph on the **intermediate PE & forward** to the **child PEs**

1. Create a **new partition** using a **graph partitioner**
2. **Send** the **new partition down the hierachy**



(3) update child PEs with the new **balanced partition**

(4) The new partition is **forwarded to the simulation PEs** so the **graph is updated**

Modifying ParHIP as HDMGP

ParHIP MGP can be integrated into HDMGP. However, changes are necessary:

1. **Modify** the **graph structure** to allow **dynamic graphs**
2. Keep **LPA for coarsening** while **modifying weight aggregation** to **leverage the hierarchical structure**
3. **Modify uncoarsening** to allocate graph **abstractions** at **each hierarchy level** while keeping the refinement algorithm (LPA)

A mechanism for **monitoring communications, workload,** and **triggering rebalancing** should be implemented. The HDMGP should run in a separate set of MPI processes using **MPI inter-communicator** operations.

Conclusion

We propose a dynamic approach for MGP using a hierarchical structure that maintains and updates the abstractions of the graph generated during coarsening. This allows for repartitioning parts of the graph when imbalances are detected.

The proposed HDMGP will run on separate PEs from the simulation, enabling repartitioning without halting the simulation's execution.

Acknowledgements

This work has been granted by the Ministerio de Ciencia e Innovación MCIN AEI/10.13039/501100011033 under contract PID2020-113614RB-C21 and PID2023-146193OB-I00 and by the Catalan government under contract 2021 SGR 00574.

UAB Universitat Autònoma de Barcelona

Escola d'Enginyeria