Load-Sharing Policies in Parallel Simulation of Agent-Based Demographic Models

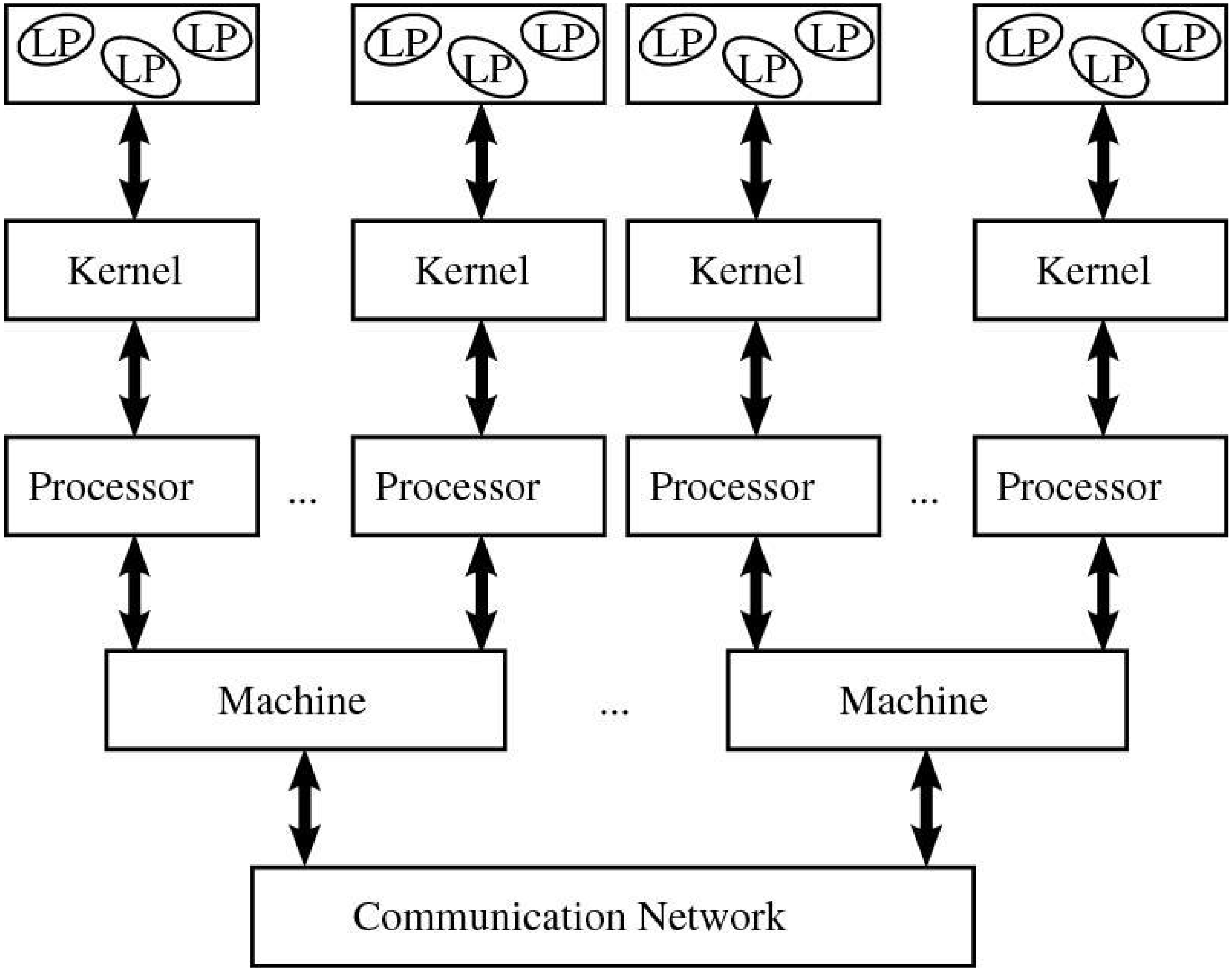
Alessandro Pellegrini
Cristina Montañola Sales
Francesco Quaglia
Josep Casanovas Garcia

PADABS 2016

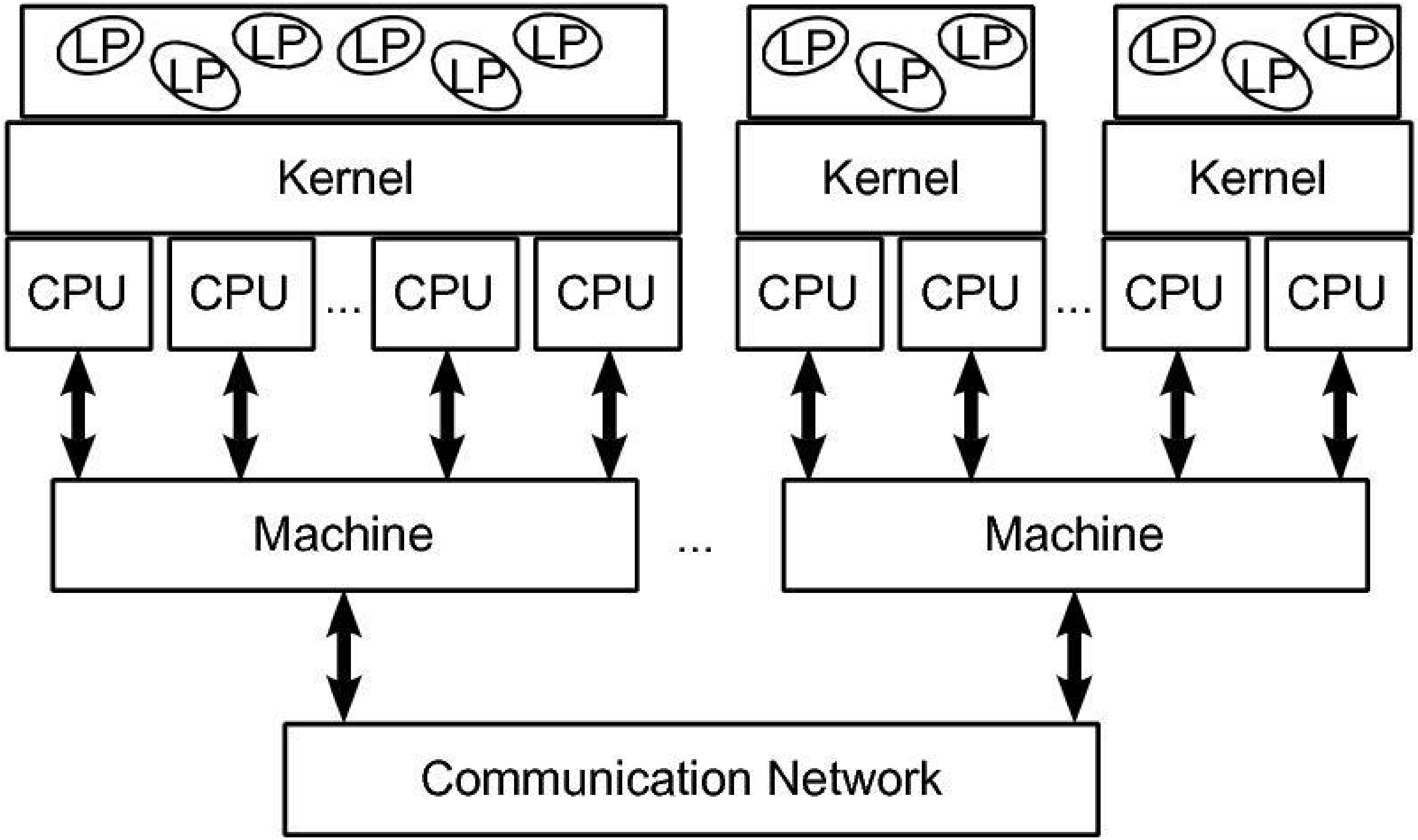
ABM, PDES & Shared-Memory Systems

- ABM is powerful thanks to its abstraction capabilities
  - Evolution of the system described through its components
  - Macro scale effects due to micro scale behaviour
  - Decision-making capabilities
  - Interaction patterns
- PDES
  - Is an effective formalism to describe Agent-Based Models
  - Entities of an AB model can be mapped to LPs
  - Many techniques to speedup the execution already exist (e.g., Time Warp)
- Shared-Memory Systems
  - Allow a significant simplification of the programming model
  - Off-the-shelf high-performance computing facilities

PDES on Distributed Systems



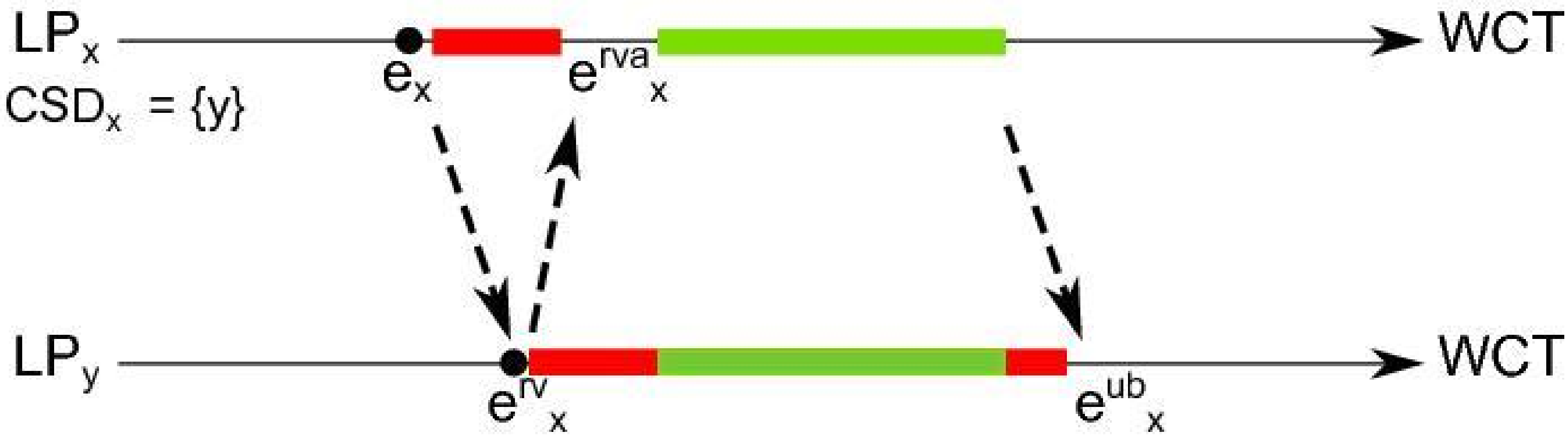
PDES on Shared-Memory Systems



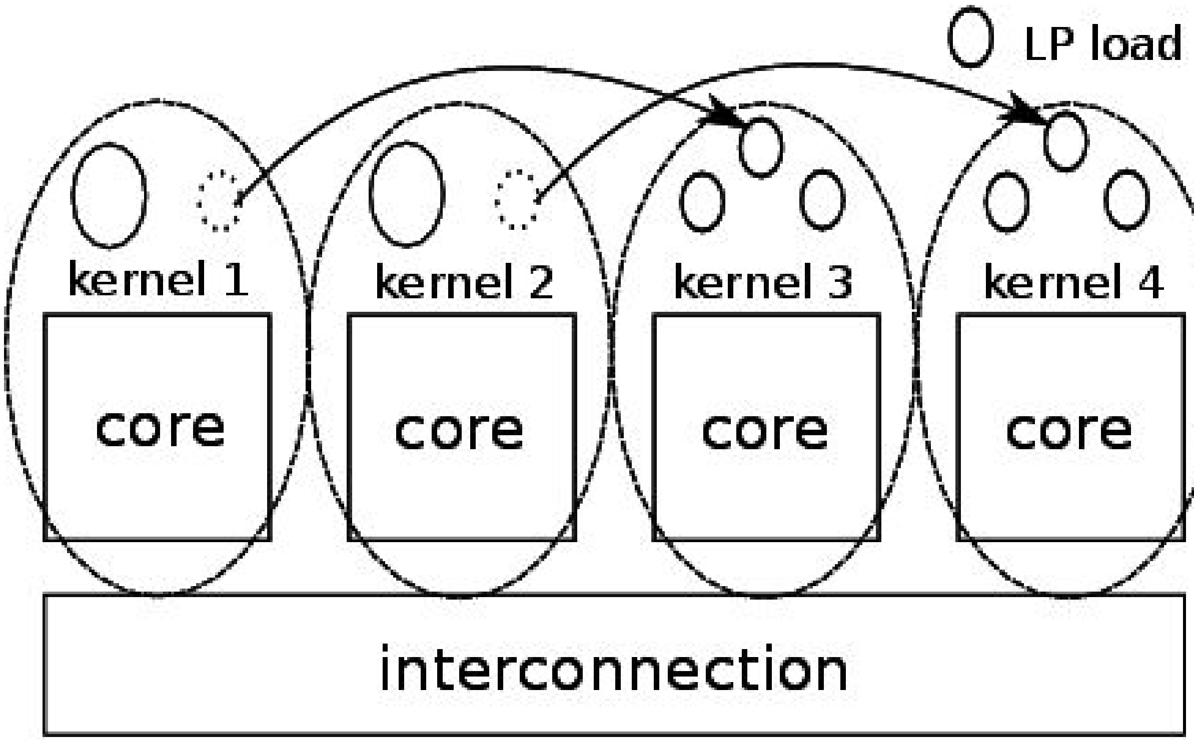
Cross-State Synchronization

- On Shared-Memory Systems, different LPs can share portions of their simulation states
  - Simplification of the programming model
  - Runtime environment must ensure consistency
- Cross State Synchronization
  - Based on OS-level facilities
  - Different threads have a different view on memory frames
  - Faulting on a different LP's page materializes a *will* to synchronize

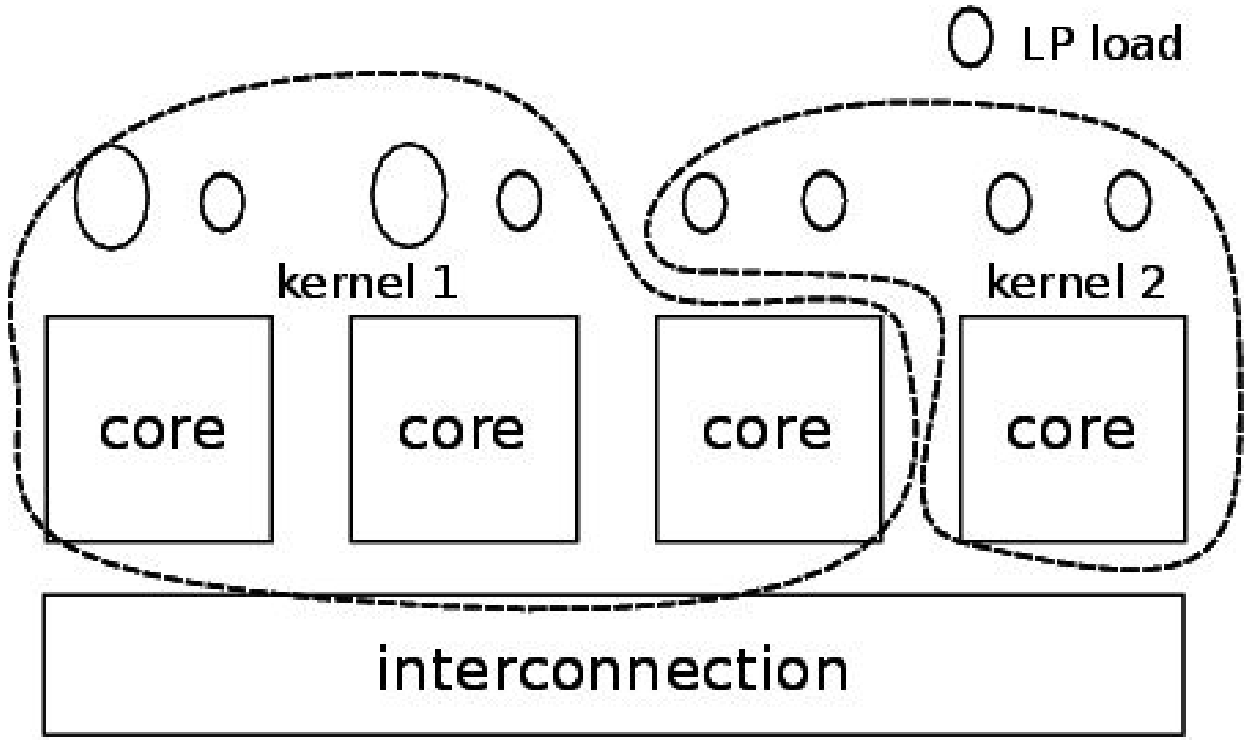
Cross-State Synchronization



The Role of Load Sharing



The Role of Load Sharing



The Role of Load Sharing

- More lightweight than load balancing
- Always use all the available computing power
- Limit the optimism of the simulation
    - Reduce the number of rollbacks
    - Increase the efficiency and scalability of the simulation
- **Main problem**: determine a proper binding between LPs and Worker Threads

Our Goals

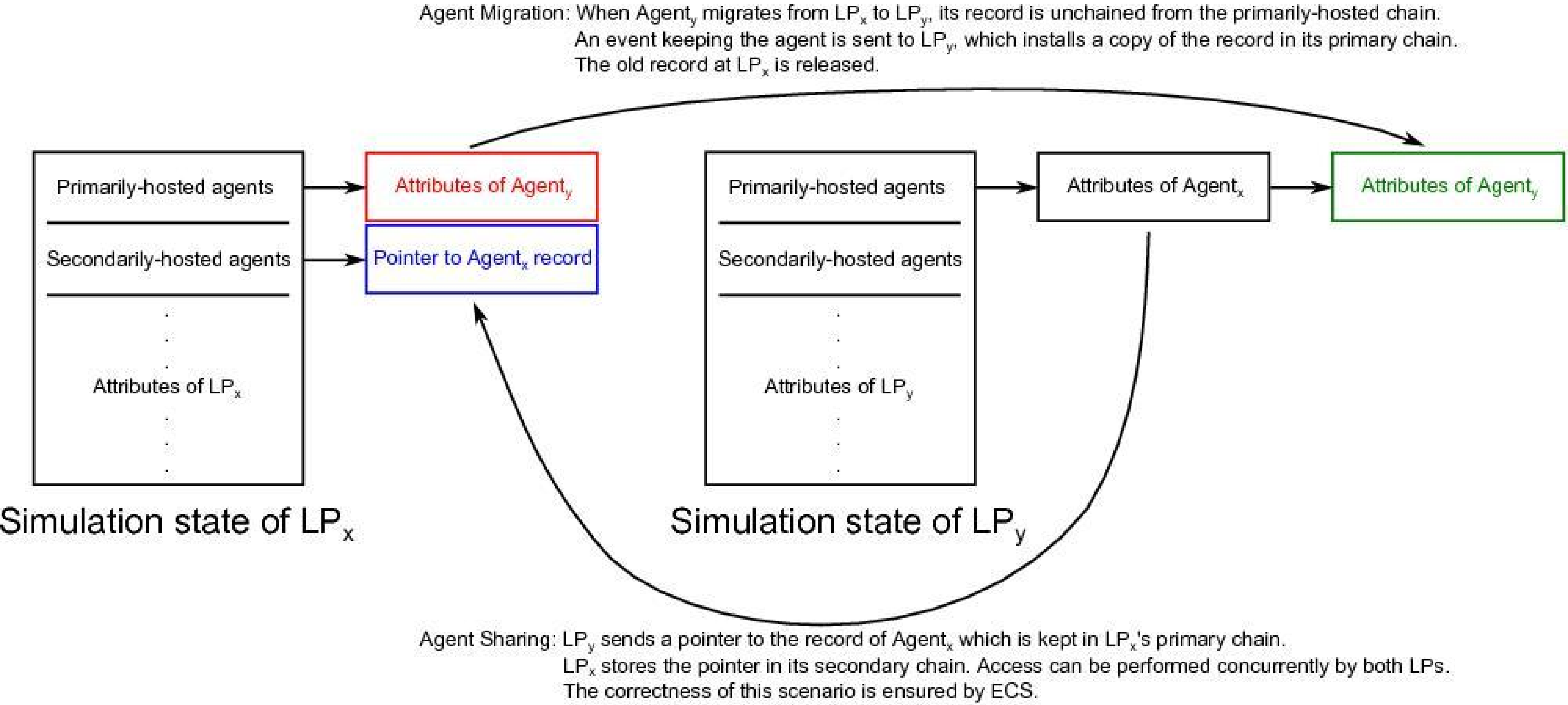
- Propose a reference programming model for AB Demographic Models using PDES on Shared-Memory Systems
  - Leverage the properties of these systems to increase programmability
  - Give the highest degree of freedom to the programmer
  - Ensure an efficient execution

- Study different Load Sharing policies
  - Consider different aspects of the simulation
    - Intercommunication
    - Future Event List density
    - Simulation advancement

Reference Programming Model

- Core elements of a demographic model
  - Life course and behaviour/decisions of individuals
  - The environment they act into
- The environment is partitioned into LPs
  - *Primary* regions: actual portions of the environment
  - *Secondary* regions: specific locations within the environment
- Agents are mapped to *data structures*
  - Individual-specific explanatory variables
  - Migrations involve moving data structures to different LPs
  - State changes are operated by LPs
- Two main events
  - Agent sharing
  - Agent migration

Reference Programming Model



Policy 1: FEL & GVT Advancement

- We consider the availability of C cores, and $K \leq C$ worker threads
- Locally, each thread k_i hosts $numLP^{k_i}$ LPs
- Each LP is associated with a workload factor:

$$L_l = \frac{q_l \cdot \delta_l}{LVT_l^{q_l} - LVT_l^1}$$

Policy 1: FEL & GVT Advancement

- These factors are aggregated in a per-thread workload factor:

$$L^{k_i} = \sum_{l=1}^{numLP^{k_i}} L_l$$

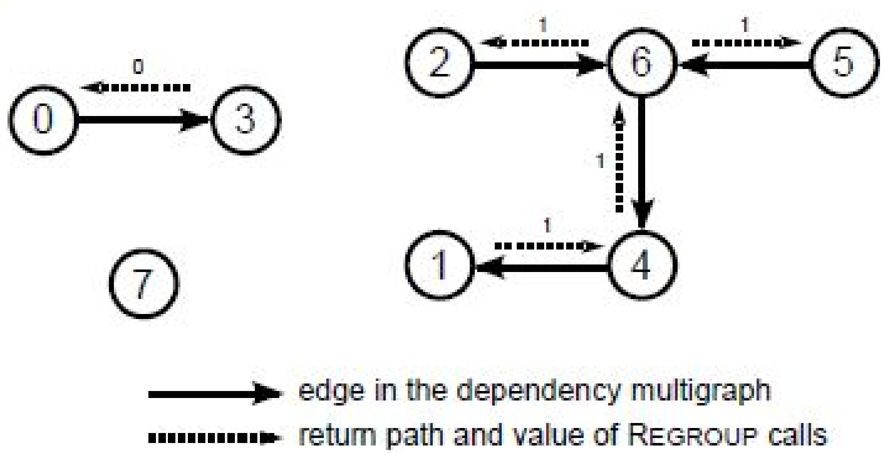
- All factors at each thread are ordered non-decreasingly
- The highest L_{l1} factor is used as a reference value
- All the other LPs are grouped together using an approximation of a *0-1 one-dimensional multiple knapsack* problem-solving algorithm

Policy 2: Implicit Synchronization

- The materialization of a cross-state access should be used to build a relation among LPs
- We use the L*pDependencies* matrix to count ECS interactions
  - *LpDependencies[i,j]* = *LpDependencies[j,i]* = #ECS interactions
- Periodically, this matrix is used to build a Directed Multigraph over the LPs
- This considers, for each LP_k, the LP_i with the highest dependency count
- A graph visiting algorithm is used to build a GLP

Policy 2: Implicit Synchronization

```
procedure REGROUP(LpGranulation GLP, int LPid, int group)
    if GLP[LPid].group ≠⊥ then
        return GLP[LPid].group
    if group ≠⊥ then
        GLP[LPid].group ← group
    else
        GLP[LPid].group ← LPid
    if GLP[LPid].MaxDep ≠⊥ then
        GLP[LPid].group = REGROUP(GLP, GLP[LPid].MaxDep, GLP[LPid].group)
    return GLP[LPid].group
```



Policy 3: Implicit and Explicit Synchronization

- Multivariable optimization problem
- We count the number of implicit/explicit synchronization activities:

$$\langle I_0, I_1, \ldots, I_{numLP-1}, E_0, E_1, \ldots, E_{numLP-1} \rangle$$

- This is a point in the n-dimensional *LPs interaction space*
- We apply the *k-medoids clustering algorithm*
- K clusters are obtained, which are mapped to worker threads

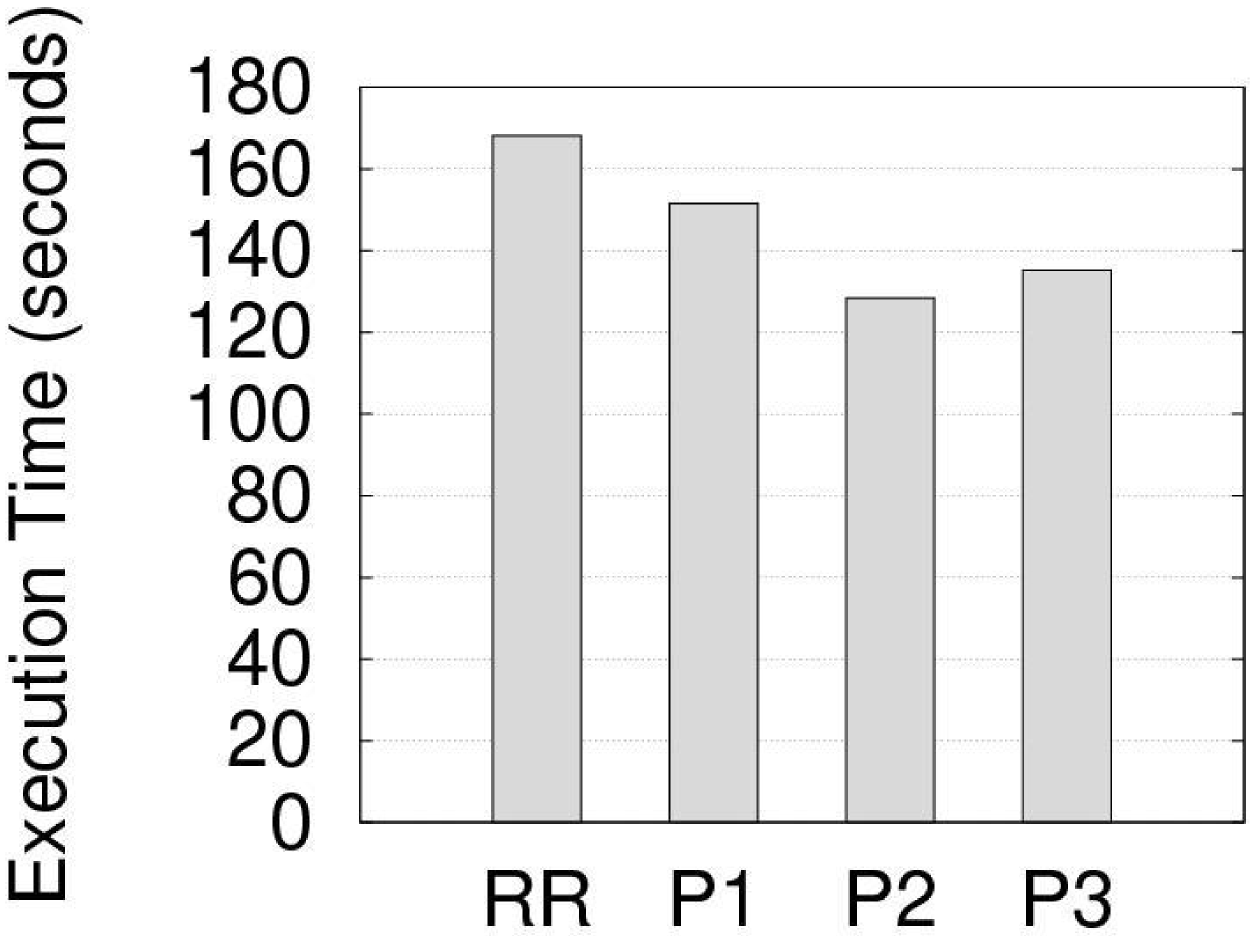
Experimental Assessment: Synthetic Model

- Each Agent has a state composed of:
  - A bitmask of attributes
  - A payload carrying less-concise information
- LPs represent both primary and secondary regions
  - When an agent enters a region, it randomly selects another LP to be shared with
- Different randomic operations:
  - *State-machine update* - some bit is negated
  - *Memory update* - some content of the payload is updated
  - *Remote agent interaction* - a message carrying data is sent to a random remote agent
  - *Agent migration* - executed after a random residence time

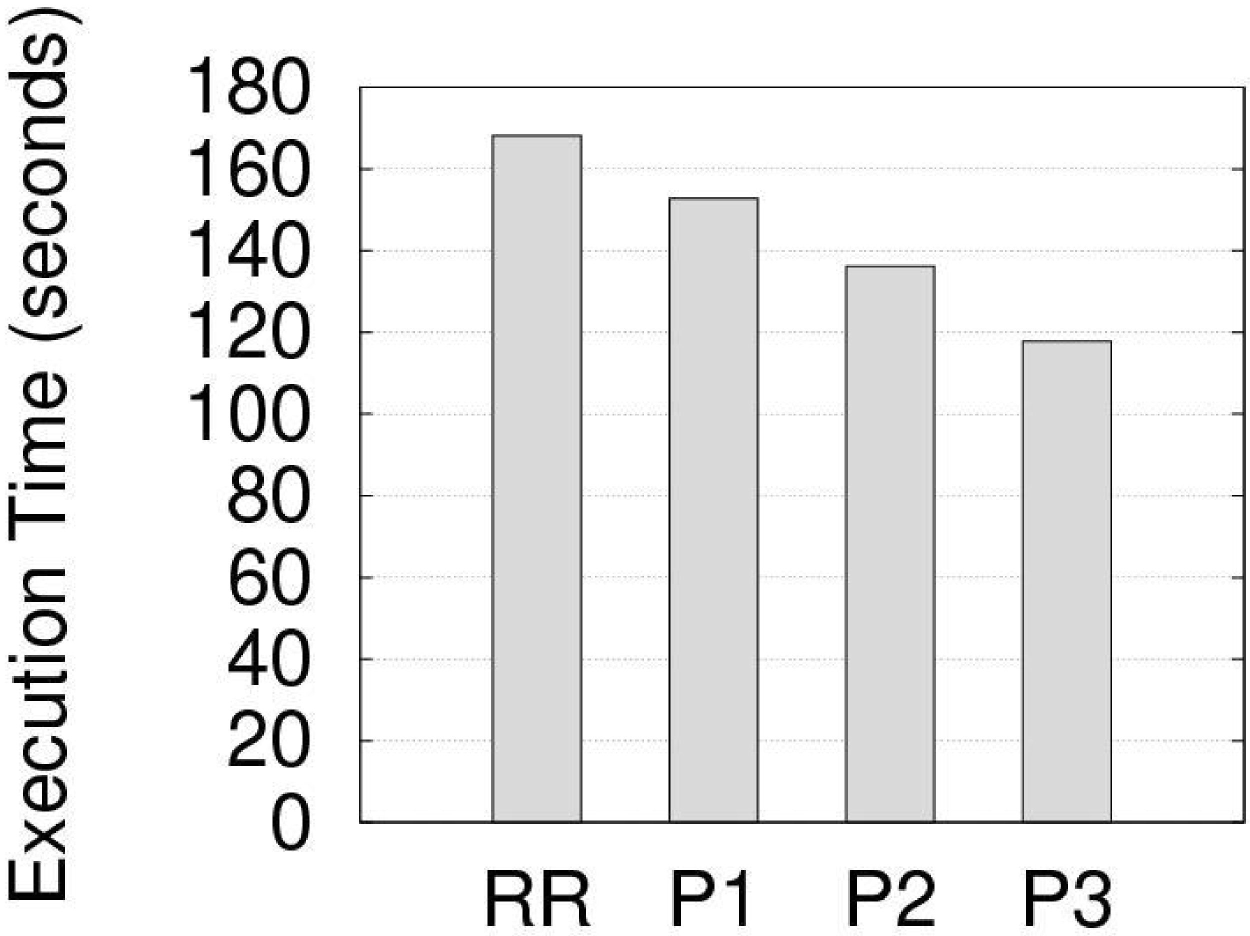
Experimental Assessment: Synthetic Model

- Operations Probabilities
  - *State-machine update*: 30%
  - *Memory update*: 50%
  - *Remote agent interaction*: 20%
  - *Agent Sharing*: 10%
- Execution using ROOT-Sim:
  - 1024 LPs (regions)
  - 100K Agents (1.6 GB of live state)
  - 32 cores
  - 32 GB of ram
- We varied the probability *p* telling whether two LPs interact via message passing

Experimental Assessment: p = 25%



Experimental Assessment: p = 75%



Conclusions

- We have discussed a parallel AB programming model for demography using optimistic PDES
  - The model allows for a sequential-style programming paradigm
  - All synchronization issues are demanded from the underlying simulation environment
- Three different load-sharing policies have been proposed
  - Load balancing is fundamental when running simulations on shared-memory machines
  - Policies which explicitly account for interactions better capture the parallelism degree of the model

Questions?