DIFFUSION-BASED LOAD BALANCING

Maya Taylor,
Kavitha Chandrasekar,
Sanjay Kale

University of Illinois at Urbana Champaign

THE TARGET RUNTIME

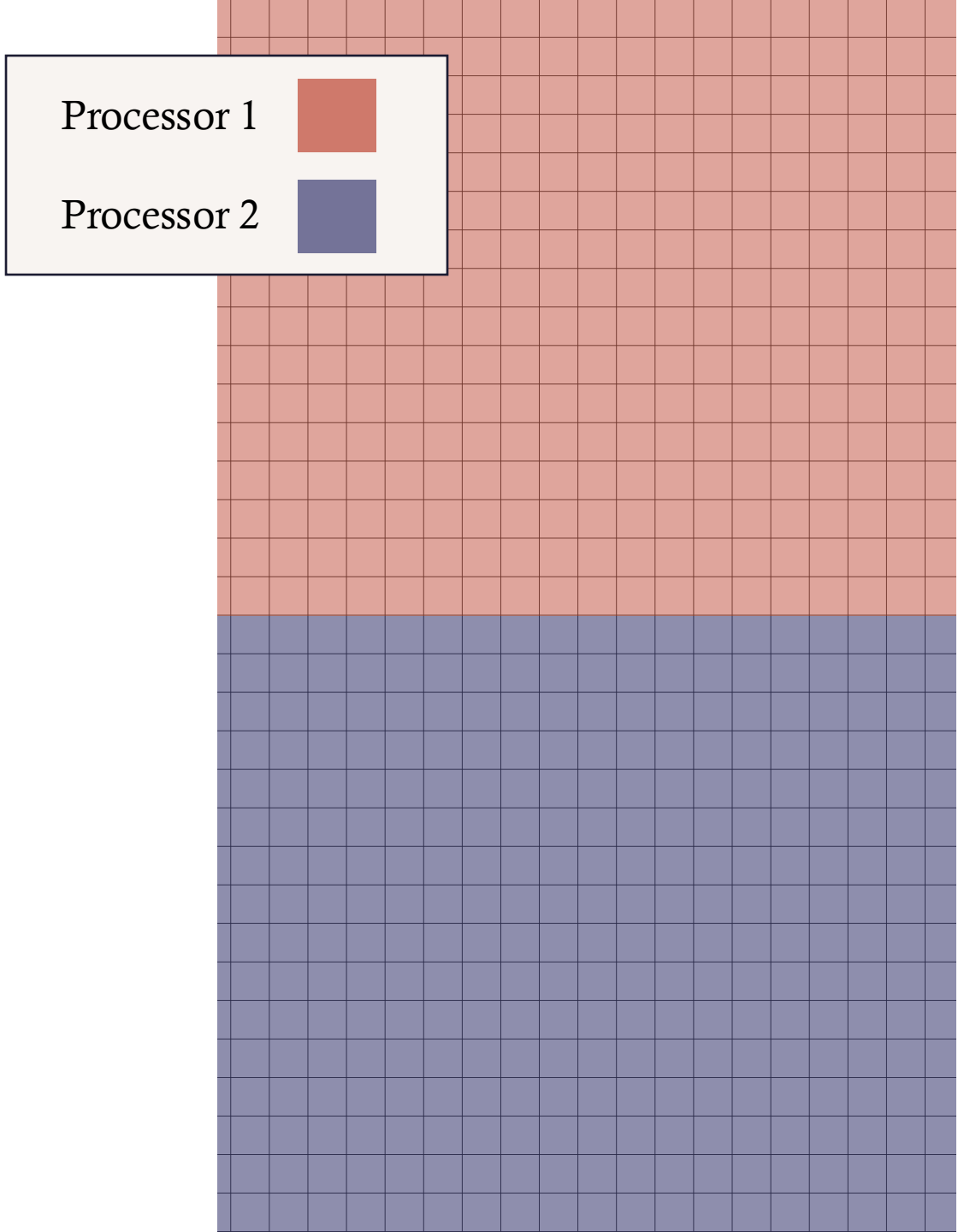
- Work is **over-decomposed** into objects
- Many objects per processor
- Objects are migratable
- Runtime can manage task migration dynamically
- E.g. Charm++

MIGRATABLE OBJECTS

- Objects have *load:* we use execution time
- Load can vary dynamically
- Objects also *communicate* with other objects
- Communication can be within a given processor, or span across processors

VISUALIZING LOAD WITH A STENCIL CODE

- Consider a stencil code
- The grid: logical application space
- Every grid element is an object containing a finer sub grid
- Objects have varying loads
- Objects communicate with their neighbors
- **Color** denotes object's resident processor

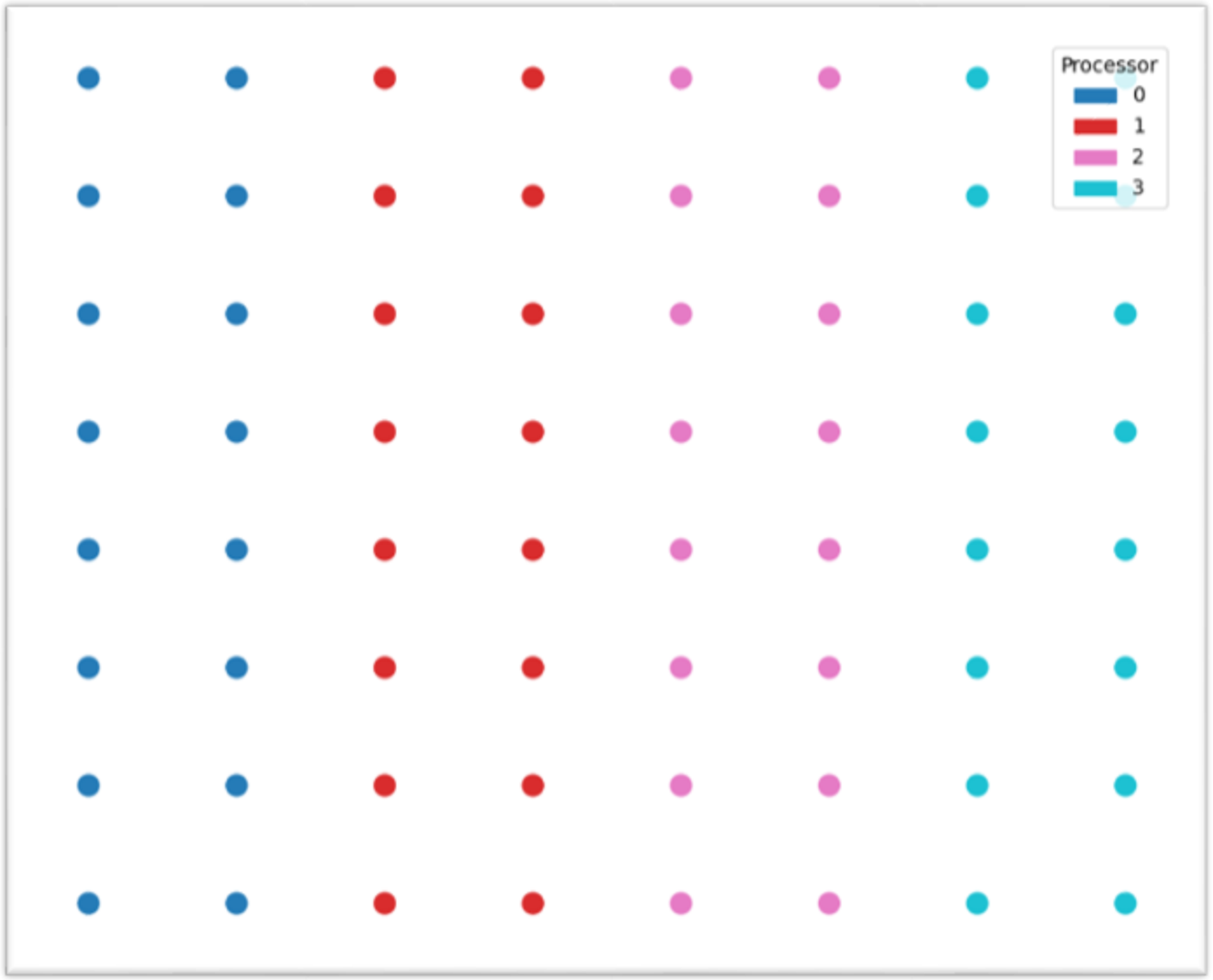


A NAÏVE STRATEGY: GREEDY REFINE

- Migrate objects from overloaded to underloaded processors
- Minimizes object migration
- Balances load effectively
- But ignores communication patterns between objects

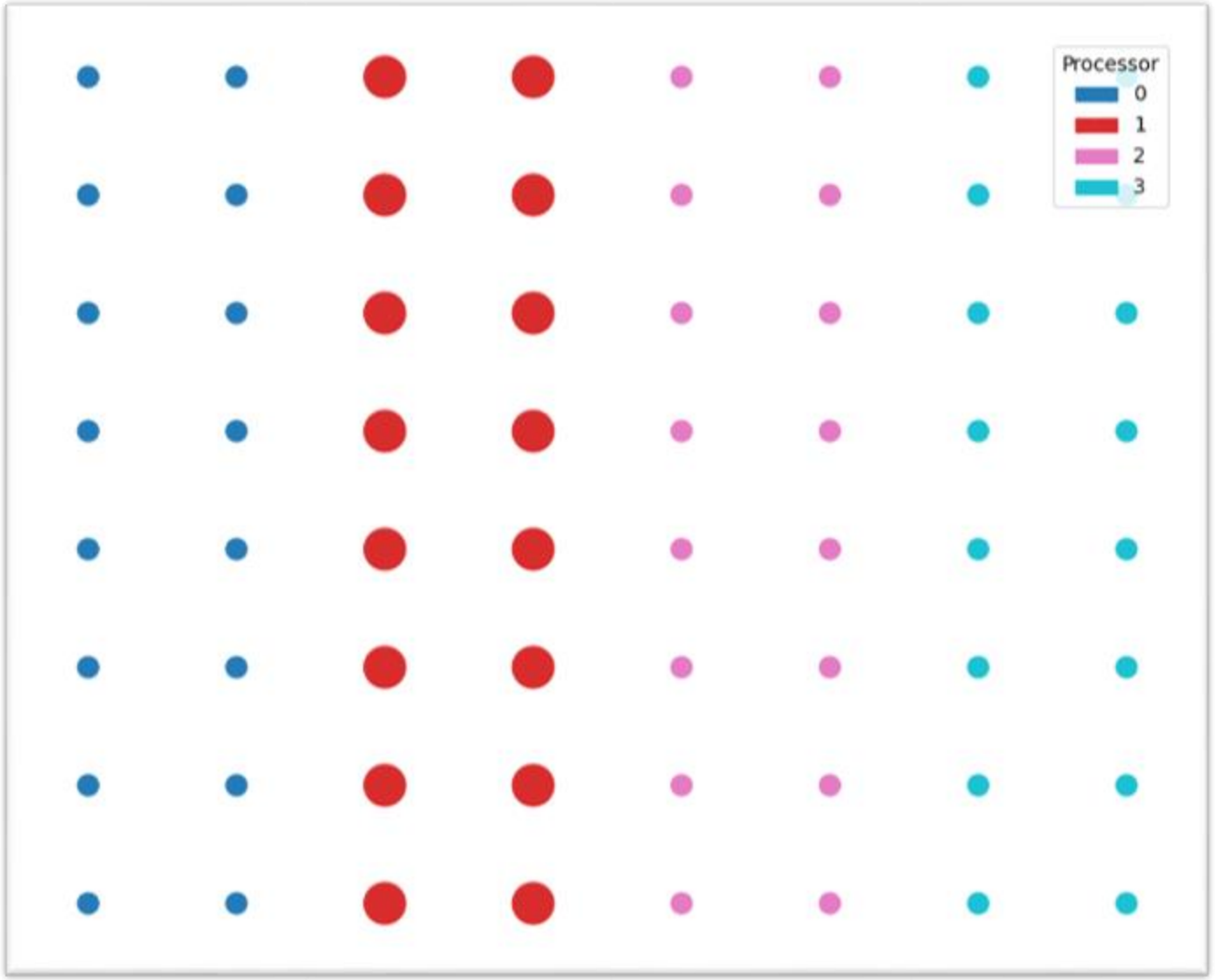
VISUALIZING GREEDY REFINE

- Stencil code
- Assume original mapping preserves communication locality



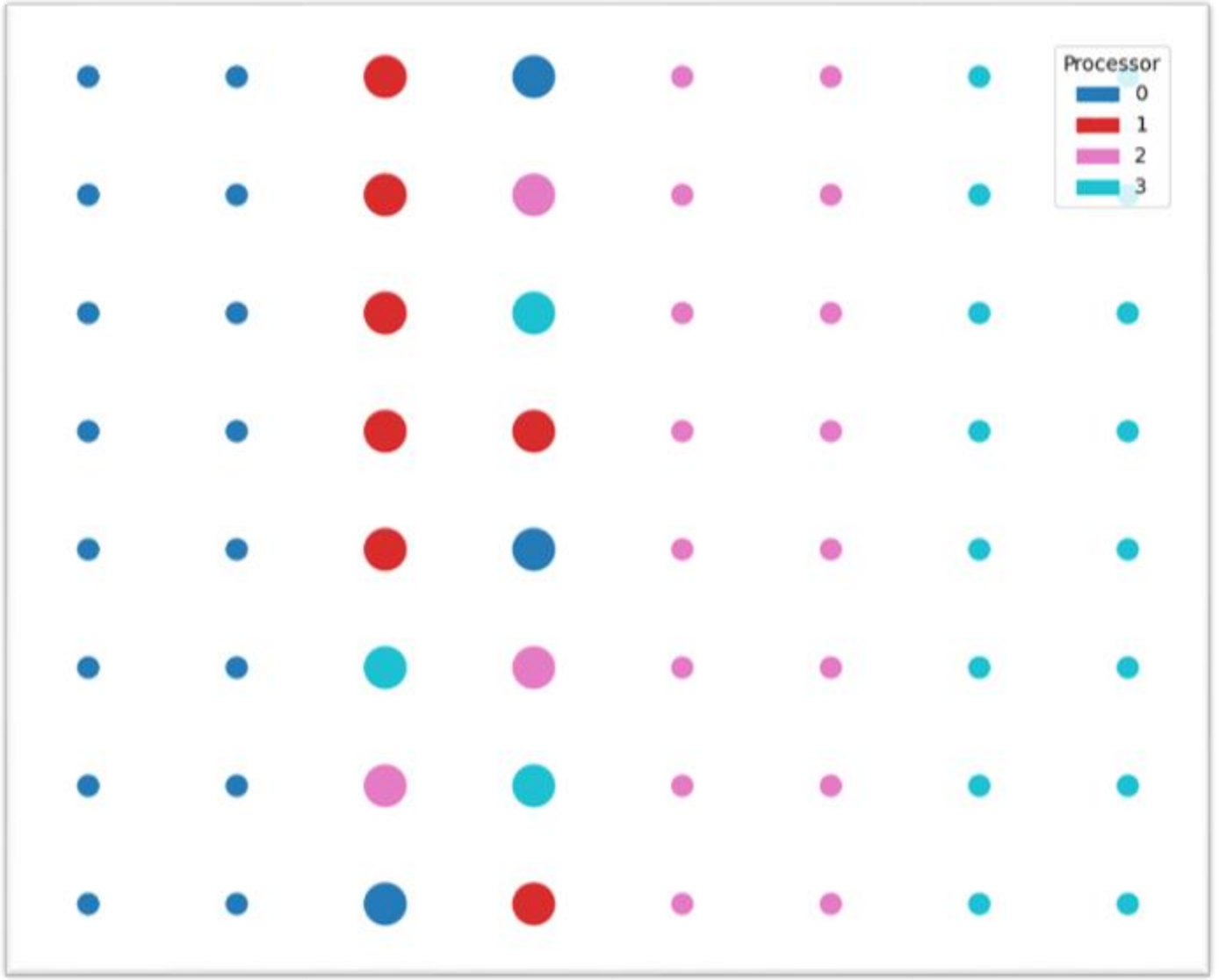
VISUALIZING GREEDY REFINE

- Inject simulated load imbalance



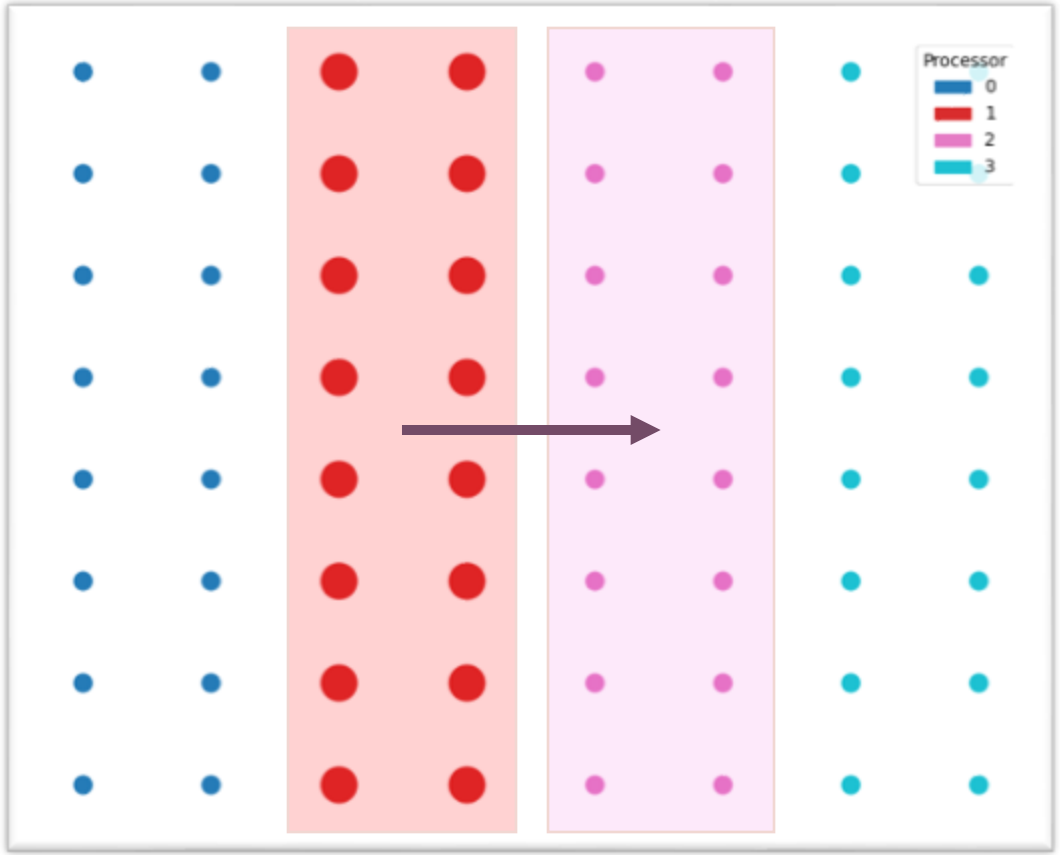
VISUALIZING GREEDY REFINE

- Migrate heavy objects
- Interspersed colors: communication locality is lost



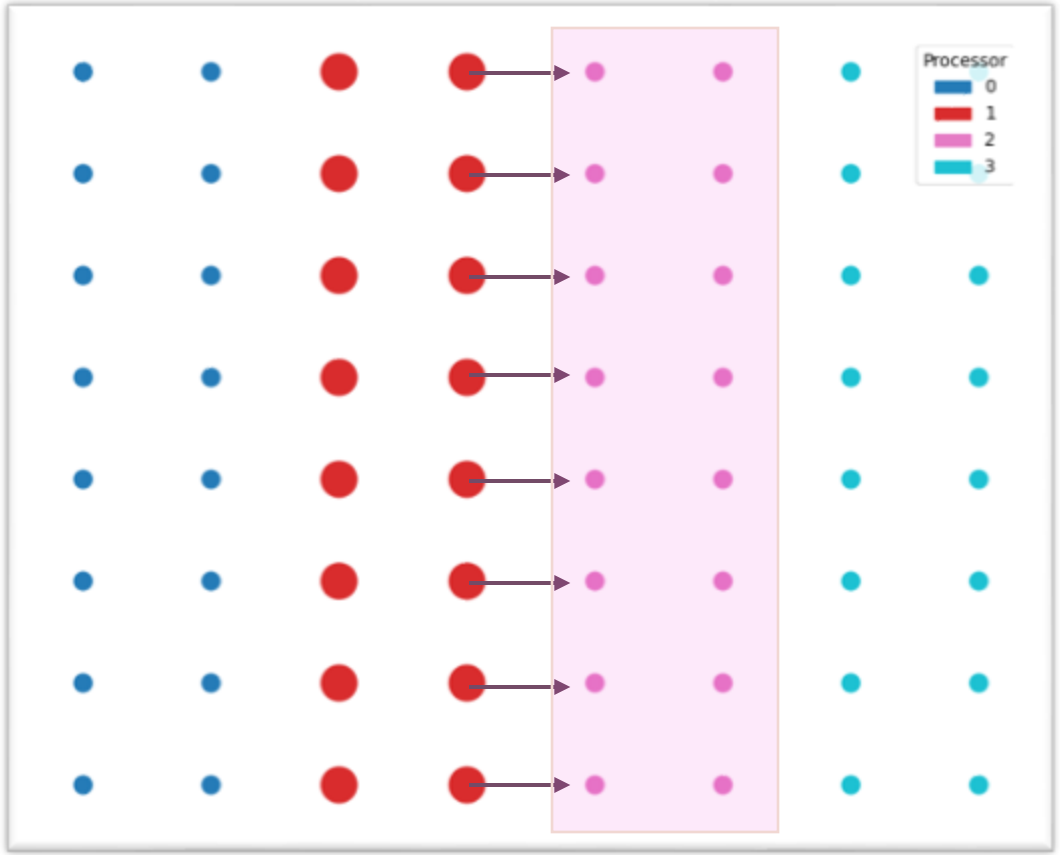
THE DIFFUSION ALGORITHM

1. Select neighbors.
   - Possible neighbors are prioritized based communication patterns
2. Determine a theoretical best diffusion pattern.
   - Each processor decides how much load to send to each neighbor
   - Uses an iterative fixed point algorithm
   - No objects are migrated – only load sizes are communicated



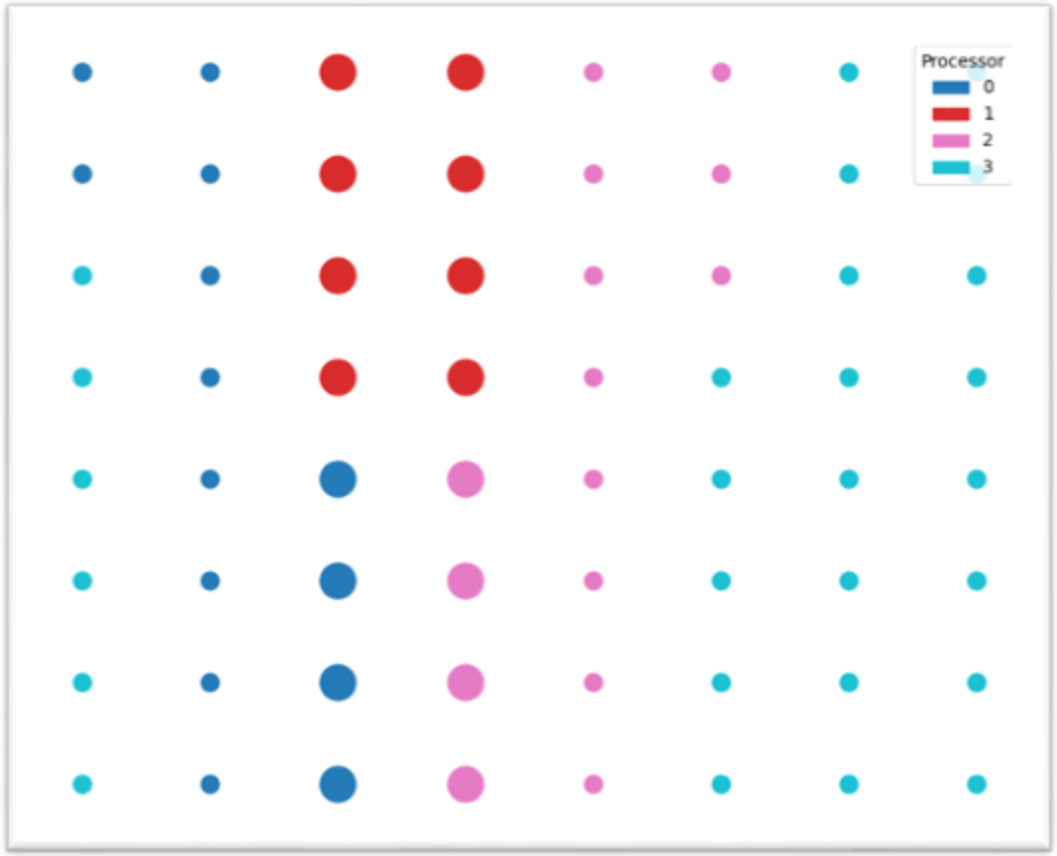
THE DIFFUSION ALGORITHM

3. Migrate objects
   - Choose objects based on existing object communication patterns
   - Attempt to meet theoretical loads computed in step 2



DIFFUSION-BASED LOAD BALANCING

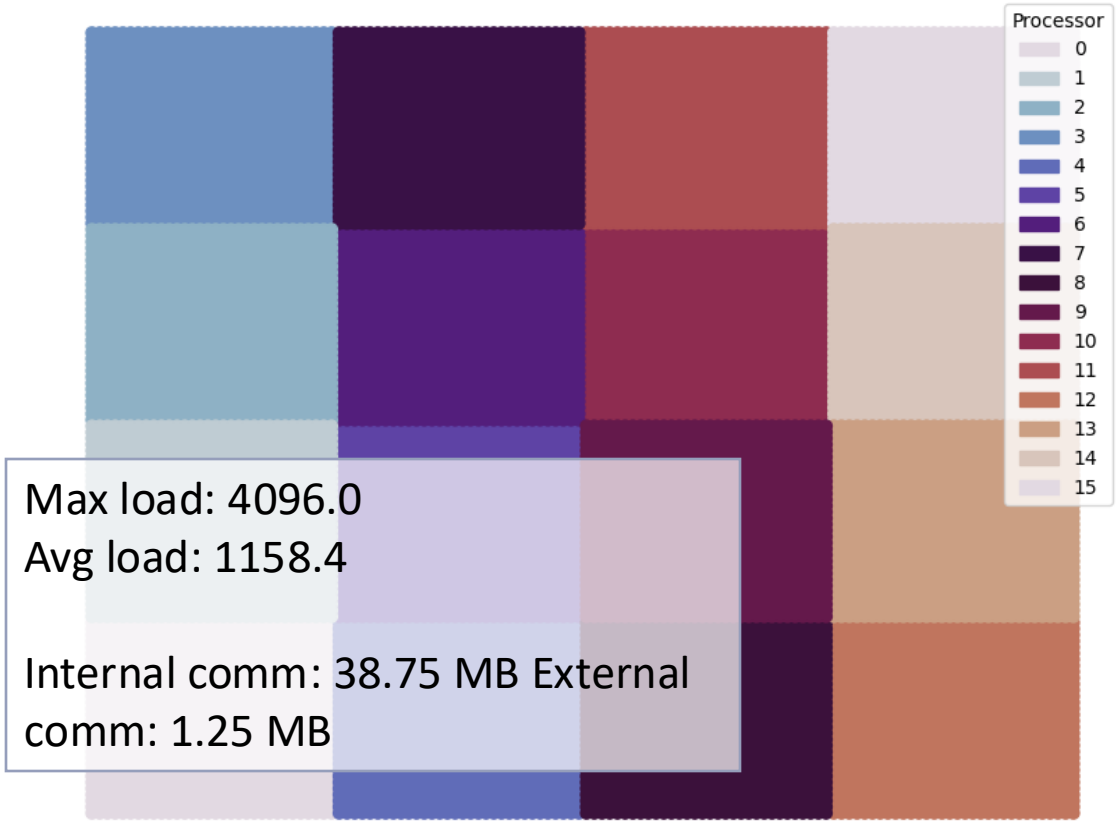
- Diffuse load along communication edges
- Preserve communication locality
- Assumes that the original mapping optimizes communication locality



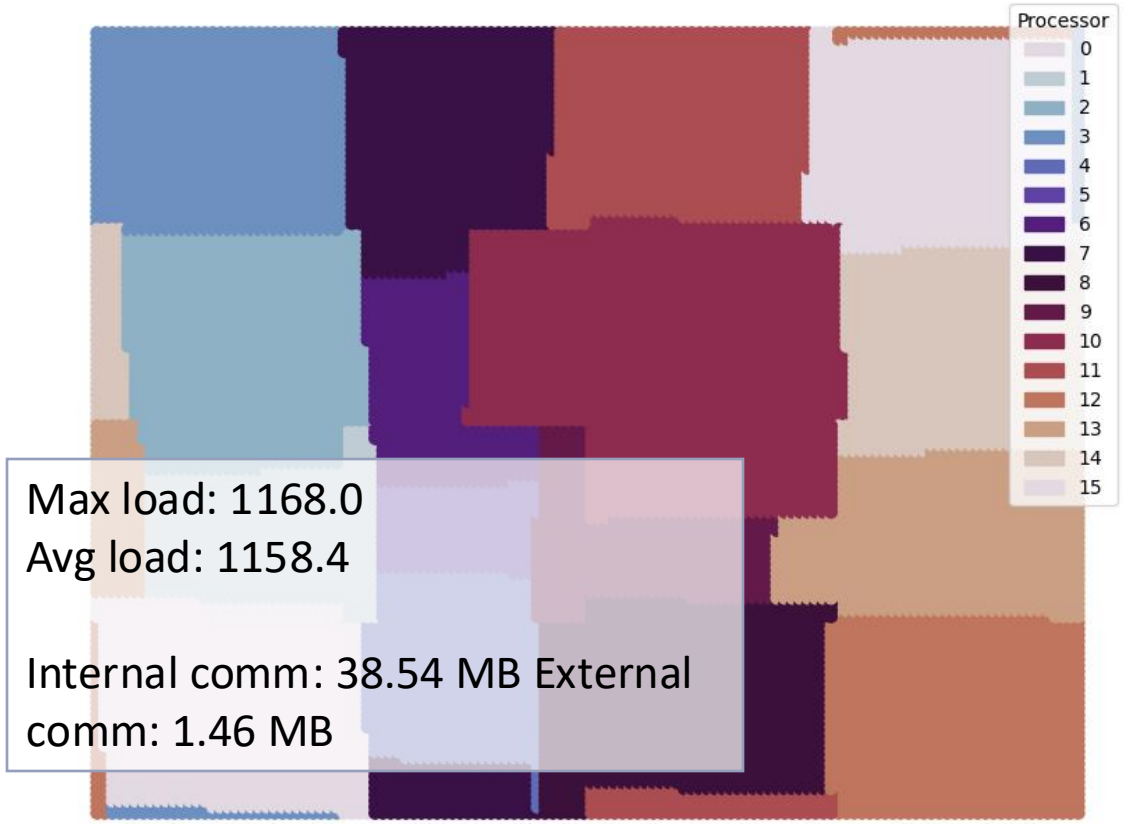
COORDINATE APPROXIMATION

- Communication graph may be
  - unavailable
  - expensive
- Use logical object coordinates to approximate communication patterns
- Assumes that proximity corresponds to communication
- Processor-level communication is approximated by the centroid coordinate of resident objects
- Neighbor processors are selected based on centroid distance

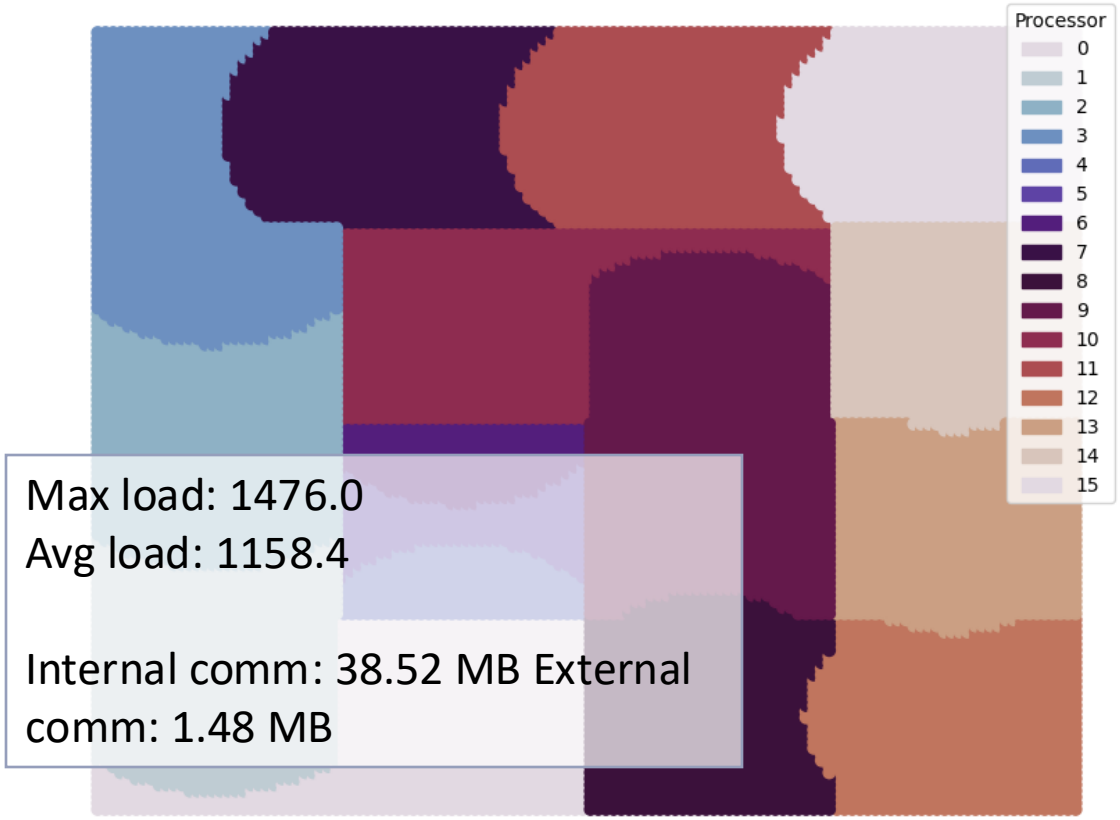
Original mapping



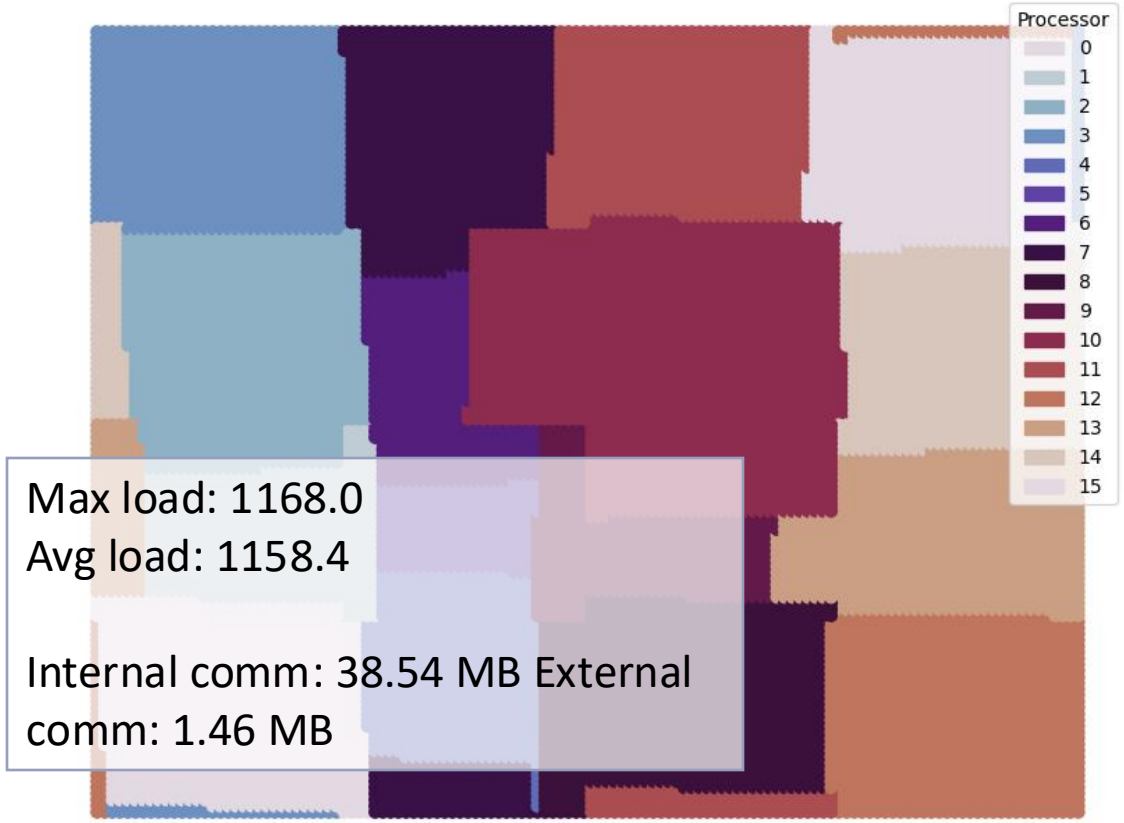
DiffusionLB (Communication)



DiffusionLB (Coordinates)



DiffusionLB (Communication)



THE LB SIMULATOR

- Charm++ framework for simulating different load balancing strategies
- Can simulate large scale LB behavior on few physical resources
- A Charm++ object represents a physical node.

FUTURE WORK

- DiffusionLB evaluation on existing Charm++ applications
  - ChaNGa cosmological simulation
- Vector load balancing
  - E.g. balancing both GPU and CPU load
- Generalizing the LB simulator

MODULARITY EFFORTS + COLLABORATION

- Goal: abstract load balancing strategies from model and runtime specifics
- Different task-based systems should be able to easily reuse each others LB strategies
- Ongoing discussion with the DARMA tasking group

- What general information should be available to a load balancing interface?
- What should the API look like?
- What visualization tools are useful and interesting?

QUESTIONS?

- DiffusionLB
- Coordinate approximation
- LB simulator and visualizations
- Current efforts towards a modular LB framework