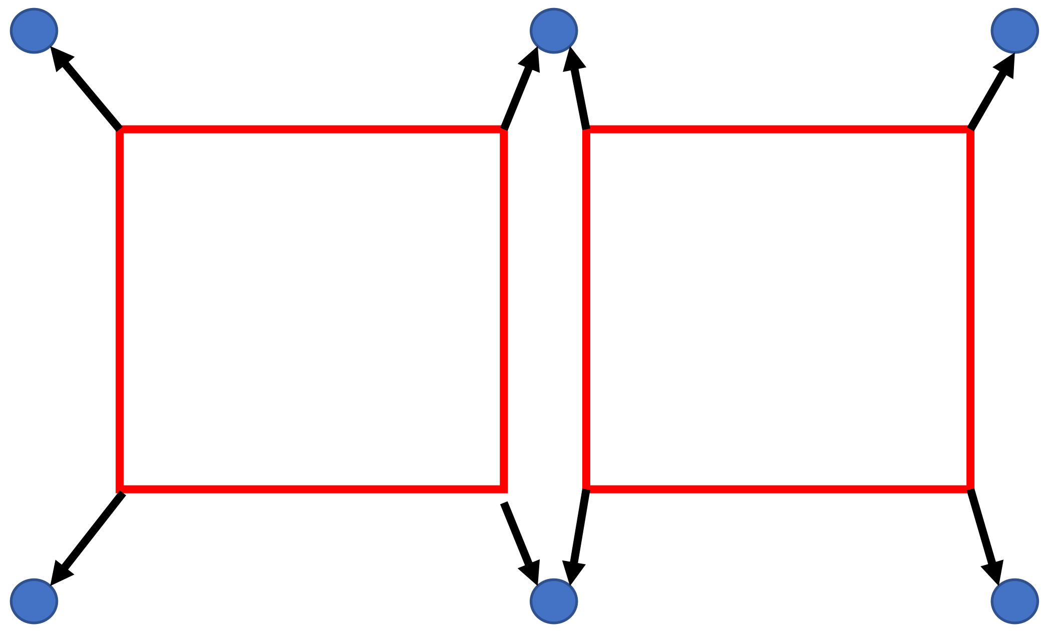
ScatterView

Scatter operation

- Like a reduction, but with many results
- Number of results is proportional to number of inputs
- Each result gets contributions from a small number of inputs
- Given inputs-to-results map, not inverse
- Examples:
    - Particles contributing to neighbors' forces
    - Particles contributing to charge in cells
    - Cells contributing forces to nodes



Scatter Algorithms

- Atomic operations
  - For each input/result pair, issue an atomic instruction to make the contribution
  - Works better on GPUs than CPUs because:
    - CPUs need to keep caches coherent
    - NVIDIA GPUs have dedicated functions for floating-point atomic add
- Data duplication
  - Each result is duplicated once for each thread of parallelism
  - For small numbers of CPU threads, can be 2X faster than atomics!
  - Uses way too much memory for large thread counts, including GPUs

ScatterView

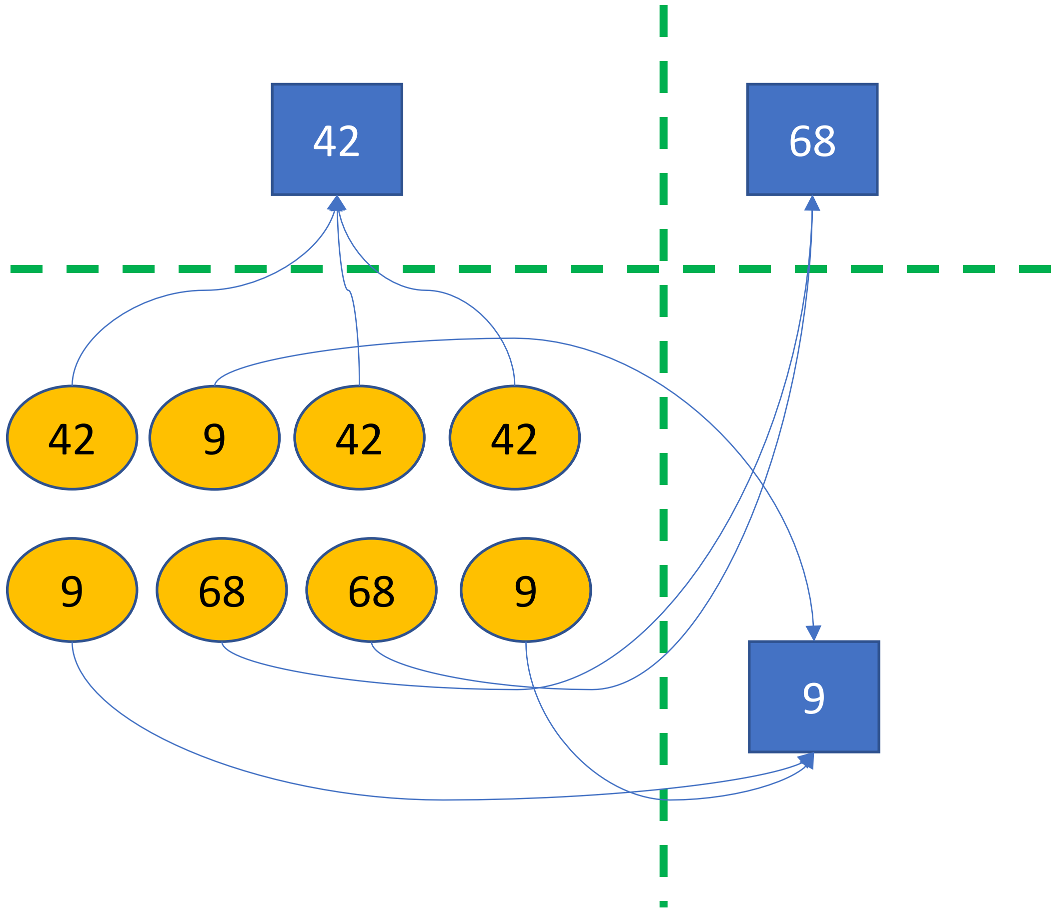
- Abstracts over different scatter algorithms
- Looks like a results view
- Only supports certain kinds of access (e.g. “+=”)
- Tuned to be as fast as hand-coded algorithms
- Part of Kokkos Containers

```
Kokkos::View<double*> results("results", 6);
Kokkos::Experimental::ScatterView<double*> scatter(results);
Kokkos::parallel_for(num_inputs, KOKKOS_LAMBDA(int input_i) {
  auto access = scatter.access();
  auto result_i = foo(input_i);
  auto contribution = bar(input_i);
  access(result_i) += contribution;
});
Kokkos::Experimental::contribute(results, scatter);
```

MPI Exchange

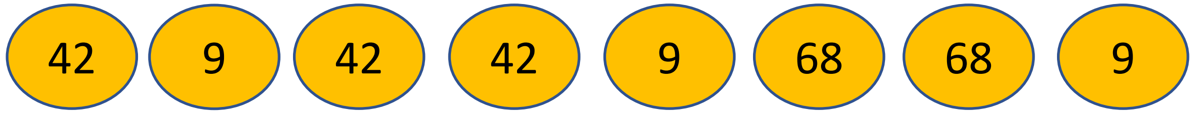
MPI Exchange

- Given lots of small "messages", each "message" has a destination rank
  - e.g. values of nodes in a mesh halo
- Assume all data is on the GPU to begin with
- Use CUDA-aware MPI to send to destinations

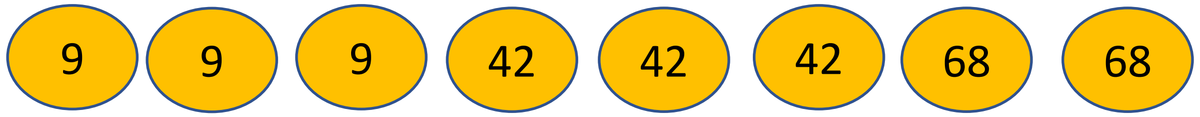


On-GPU Exchange Steps

1. Sort by destination rank, get permutation



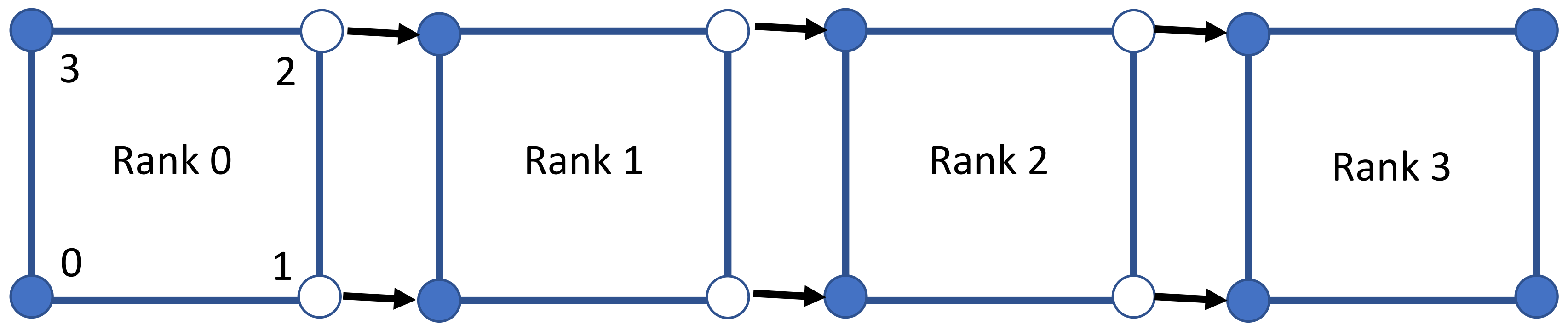
2. Determine number of “messages” received
3. Permute message data



4. Pass to MPI_Neighbor_alltoallv

Example

- Summing contributions from non-owned nodes to owned nodes



● Owned

○ Not Owned