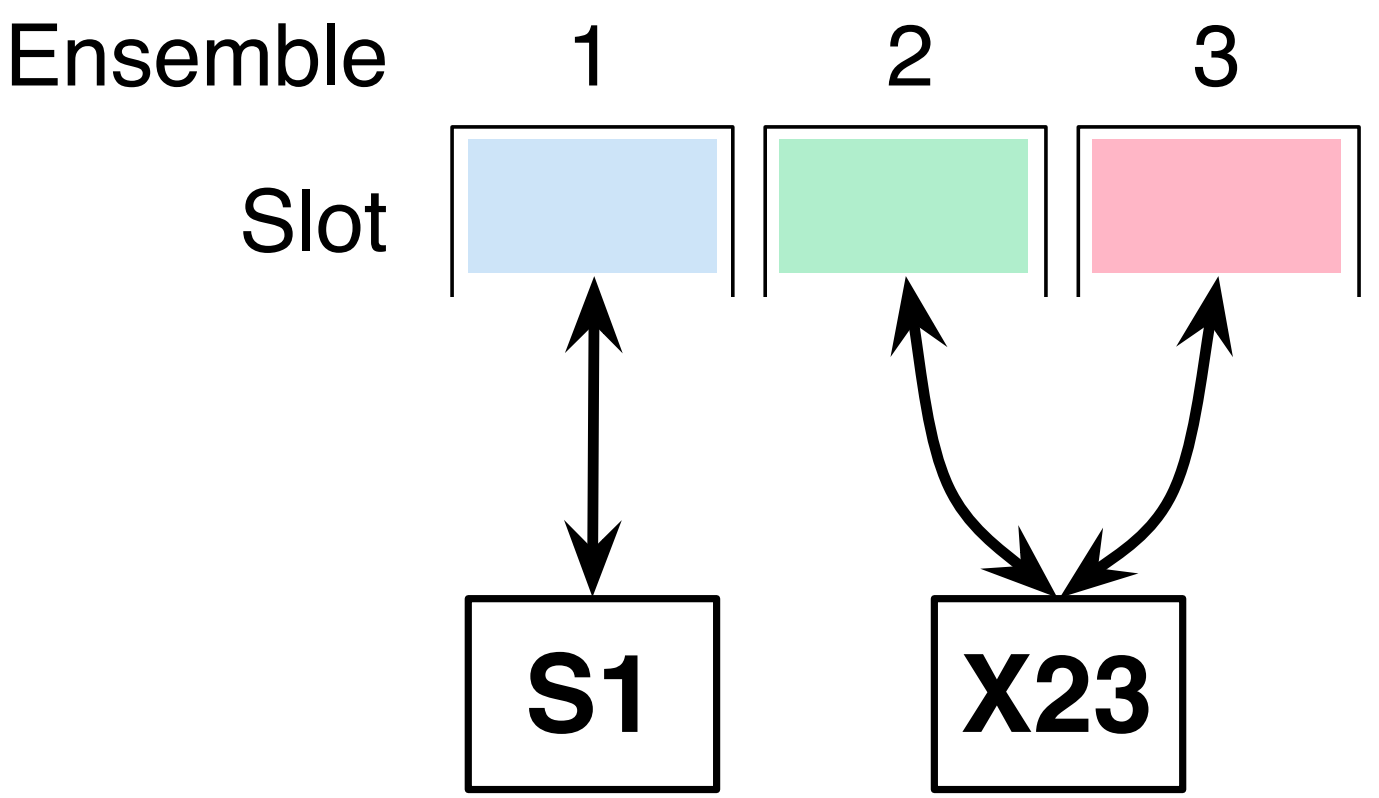
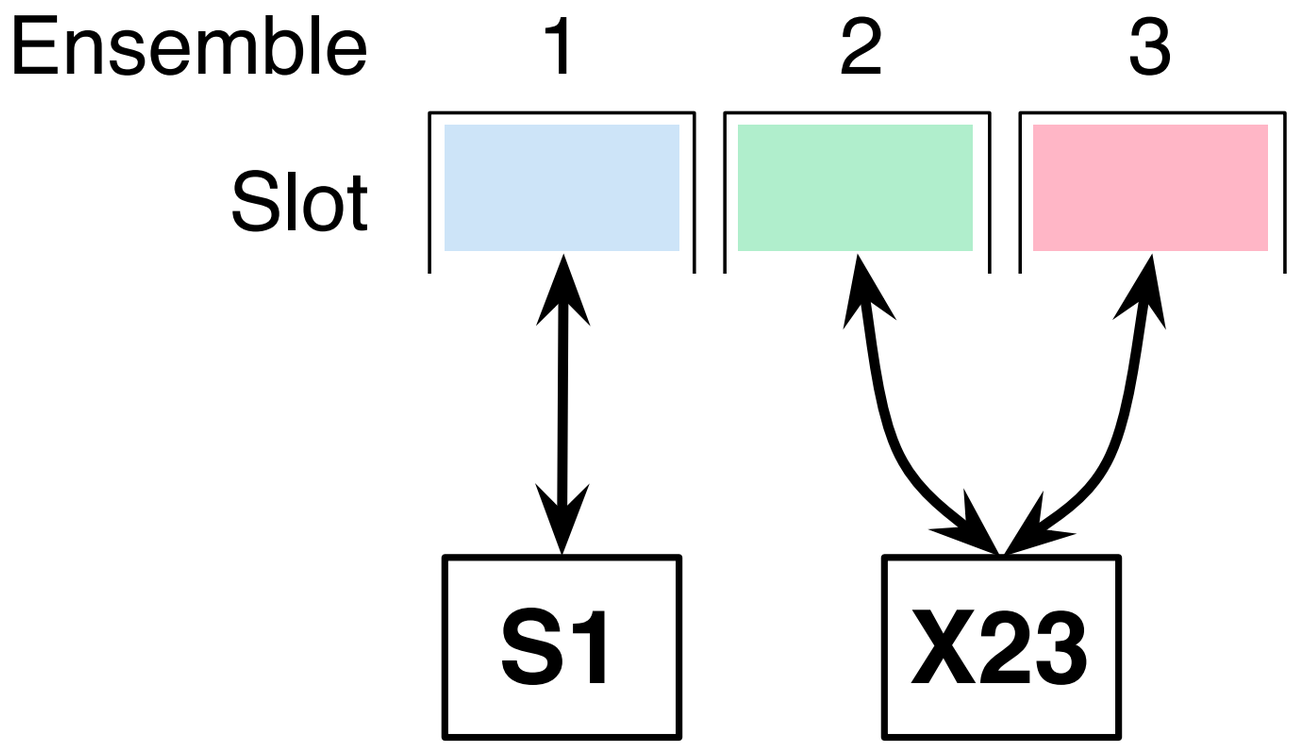
Algorithm:

1. Pull relevant trajectories from ensemble slots
2. Do the Monte Carlo move
3. Update the ensemble slots with the results


Ensemble
1
2
3
Slot
S1
X23



Algorithm:

1. Pull relevant trajectories from ensemble slots
2. Do the Monte Carlo move
3. Update the ensemble slots with the results

Dask makes task graphs easy!

Find the parts of your code that can be delayed for later execution

Wrap those functions with dask (delayed or futures API)

Inputs and outputs of functions determine task graph

```
def do_mc_step(mover, s_ensemble_slots):
    s_mover = serialize_wrap(mover)
    input_ensembles, output_ensembles = mover.ensemble_signature
    all_ensembles = set(list(input_ensembles) + list(output_ensembles))

    # select input samples
    input_s_slots = [s_ensemble_slots[get_uuid(ens)] for ens in input_ensembles]
    d_samples = dask.delayed(select_samples_from_slots, pure=False)(input_s_slots)

    # run the task
    d_change = dask.delayed(move_task, pure=False, nout=2)(s_mover, d_samples)

    # update the slots
    for ens in all_ensembles:
        updated_slot = dask.delayed(update_slots)(d_change, d_samples, s_ensemble_slots[get_uuid(ens)])
        s_ensemble_slots[get_uuid(ens)] = updated_slot

    return d_change, s_ensemble_slots
```