

```

Data :  $M, POsToMove$ 
Result: migrate partition objects in  $POsToMove$ 
begin
  /* STEP 1: collect entities to process and clear partitioning data. See
  §4.4.1 */
  for each  $M_i^d \in POsToMove$  do
    insert  $M_i^d$  into  $entitiesToUpdate[d]$ ;
    reset partition classification and  $\mathcal{P}$ ;
    for each  $M_j^q \in \{\partial(M_i^d)\}$  do
      insert  $M_j^q$  into  $entitiesToUpdate[q]$ ;
      reset partition classification and  $\mathcal{P}$ ;
    endfor
  endfor
  /* STEP 2: determine residence partition. See §4.4.2 */
  M.setResidencePartition( $POsToMove, entitiesToUpdate[q]$ );
  /* STEP 3: update partition classification and collect entities to remove.
  See §4.4.3 */
  for  $d \leftarrow 3$  to 0 do
    for each  $M_i^d \in entitiesToUpdate[d]$  do
      determine partition classification;
      if  $P_{local} \notin \mathcal{P}[M_i^d]$ 
        insert  $M_i^d$  into  $entitiesToRemove[d]$ ;
      endif
    endfor
  endfor
  /* STEP 4: exchange entities. See §4.4.4 */
  for  $d \leftarrow 0$  to 3 do
    M.exchangeEnts( $entitiesToUpdate[d]$ );
  endfor
  /* STEP 5: remove unnecessary entities. See §4.4.5 */
  for  $d \leftarrow 3$  to 0 do
    for each  $M_i^d \in entitiesToRemove[d]$  do
      if  $M_i^d$  is on partition boundary
        remove copies of  $M_i^d$  on other partitions;
      endif
      remove  $M_i^d$ ;
    endfor
  endfor
  /* STEP 6: update ownership. See §4.4.6 */
  for each  $P_i^d$  in  $P$  do
    owning partition of  $P_i^d \leftarrow$  the poorest partition among  $\mathcal{P}[P_i^d]$ ;
  endfor
end

```

**Algorithm 4.2:** M.migrate( $M, POsToMove$ )

**Note:**

- Term 'partition' in Algorithm 4.2 means 'partition' or 'part' in ITAPS parallel interface terms.
- There are 6 steps in the migration algorithm, and 5 steps of them are involved with communications (except **STEP 3**).

**Algorithm Input Data:**

- **M**, a distributed mesh with partition model **P** stored in each part. Entity's partition classification and remote copies are pre-computed.
- **POsToMove**, a list of partition objects to migrate and their destination part ids.

**Result:** Migrate partition objects in *POsToMove*

- **STEP 1:** Collect entities to process and clear their partition classifications.

In **STEP 1**, the entities collected in *entitiesToUpdate* include:

- $M_i^d \in \text{entitiesToUpdate}$ ;
- For each  $M_i^d$ , all downward entities that bound  $M_i^d$ , and their remote copies on other parts.

If an entity on one part is in *entitiesToUpdate* and has remote copies, its remote copies on other parts should also be collected into *entitiesToUpdate* through one round of communication.

- **STEP 2:** determine residence parts where each mesh entity in *entitiesToUpdate* will exist after migration.

Firstly, the residence parts for the entities in *entitiesToUpdate* are decided on each part. Then one round of communication is performed to exchange residence parts of entities on the part boundaries to unify them between remote copies.

- **STEP 3:** update partition classification for each  $M_i^d \in \text{entitiesToUpdate}$ . For each  $M_i^d$ , if its residence parts does not include its current local part, collect it into *entitiesToRemove*.

No communication.

- **STEP 4:** exchange entities in *entitiesToUpdate* from dimension 0 to 3, and update remote copies.

This step requires at least 3 rounds of communications: sending messages to destination parts to create entities, then destination parts sending back new entities created, finally allowing all copies know other remote copies.

In each round of communication, different copy of an entity may control the flow of messages.

- **STEP 5:** remove unnecessary entities in *entitiesToRemove* from dimension 3 to 0.

If a mesh entity to remove is on the part boundary, its remote copies on other parts must be removed through one round of communication.

- **STEP 6:** update the ownership for each partition model entity  $P_i^d$  stored in the partition model  $\mathbf{P}$ , based on poor-to-rich ownership rule.

One round of communication is performed to collect the number of partition objects on each part, and then the ownerships are decided based on the number of partition objects.