Data Placement Reloaded

Contents

Setting

Every Hermes system instance includes one or more Hermes *nodes*.

The storage resources set aside by the user for Hermes are *tiered*.

A *destination* is a buffering resource that can be identified by a pair of node + tier "coordinates".

Each destination d_k has characteristics such as the following:

- A capacity $Cap[d_k]$
- A remaining capacity $Rem[d_k]$
- A speed (or throughput) $Speed[.,d_k]$
  - This is the mean of the throughputs of all ranks associated with the destination's node
  - **Fix this!** Speed is really a function of the origin.
- ...

Note: At any point in time, there's a degree of *uncertainty* to some of the destination characteristics. For example, the remaining capacity of a destination is typically obtained from a global MD structure that is updated asynchronously. Only the Hermes node buffer pool managers have the precise value(s) for the pool under their management.

Problem

Input:

- MPI rank
- Vector of BLOBs $(b_1, b_2, \ldots, b_B)$
- List of destinations $(d_1, d_2, \ldots, d_D)$
    - $d_k := (Node[d_k], Tier[d_k], Cap[d_k], Rem[d_k], Speed[d_k], \ldots)$
    - **Note:** Typically, the destination list will only include a *small* fraction of all destinations in a Hermes instance.

Output: Vector of buffer IDs $(\beta_1, \beta_2, \ldots, \beta_P)$ where

- $\sum_{1 \leq i \leq B} b_i = \sum_{1 \leq j \leq P} \beta_j$ (placement in full) and
- $\forall 1 \leq j \leq P \exists 1 \leq k \leq D \; \beta_j \in d_k$ (buffers conforming to the list of destinations)

Solution

1. Pick a DP solver to obtain a *tiered schema*
    - Linear programming
        - Constraints
        - Objective function
    - Round-robin
        - Granularity
    - Random
        - Distribution(s)
2. Use the buffer pool's "coin selector" to convert into buffer IDs
3. Handle two types of potential errors
    - DP solver failure: This can happen because of insufficient capacity, constraint infeasibility, etc.
    - Coin selection failure: This can happen because of outdated state view information, e.g., outdated remaining capacities.

Error Handling

In both cases, the list of destinations is inappropriate and needs to be updated or changed.

The list of "relevant destinations" for a rank is assembled by the Hermes node *topology generator*. It gets triggered when DP fails. The initial topology consists of "node-local" destinations (Plan A) plus a backup list of neighbors (Plan B) to consult when a rank gets in trouble. If both plans fail, the topology generator invokes the *application-level* "rebalancer" to redraw neighborhood boundaries. (Plan C) In the past, we used to call these components node- and application-level DPEs, but they aren't directly involved in DP decisions, and we need maybe a clearer terminology.

Retrieved from "https://hermes.page/index.php?title=Data_Placement_Reloaded&oldid=393"

This page was last edited on 19 August 2020, at 15:59.