**VM Cloud VM pool and batches maintenance design**

Internal states

* The pool: collection of VM id’s that are created
* VM data: Map from VM id to information about the VM, i.e. OpenStack related info, VNC password, IP address, etc.
* Batch list: A list of batches, each containing the batch ID, remaining batch size, and VMs that are ready and assigned to this batch
* Preparation queue: A queue of batch IDs each signifying that a VM needs to be prepared for the batch
* VM handle list: Map from VM handle to its state and VM id assignment
* Pending locks list: List of pending locks each containing the VM handle and the batch ID

API to expose to application server & Implementation

* Set pool size constraints (min pool size, max pool size, linger time)
* Min size: #VMs to always maintain
* Max size: Max #VMs running
* Linger time: amount of time VM spends being free before it considered killable
* Rules:
  + Whenever #VMs < min pool size
    - Boot a VM
  + Whenever #VMs > min pool size and there is a VM #X that is free and has been free for [linger time]
    - Shutdown VM #X
  + Whenever #VMs > max pool size and there is a VM #X that is either FREE or BOOTING (BOOTING takes priority)
    - Shutdown VM #X
* Prepare batch (batch size, data) -> batch id
* Batch size: # workers to work on this task = # VMs to prepare
* Data: initialization data, such as homepage
* Rules:
  + Upon this call, the tuple (batch id)\*batch size is added to the preparation queue; (batch id, batch size, data, VMs:[ ]) is added to the batch list.
  + Whenever the preparation queue is non-empty and there is a free VM #X
    - Pop batch id #Y from the preparation queue; prepare VM #X with batch data of #Y; append #X to the list of VMs of batch #Y
      * When preparation is done, check if #X is still in the list of VMs of batch #Y. If not, clean up #X.
  + Whenever there is a VM #Y in batch #X that no longer exists or is in the ERROR state
    - Remove #Y from batch #X and add #X to the preparation queue
  + Whenever the preparation queue is non-empty, there is no free VM, the #VMs < max pool size, and # preparation queue items > # VMs in CREATING/BOOTING state
    - Boot VM #X
  + Whenever the preparation queue is non-empty, there is no free VMs, and the #VMs >= max pool size
    - Do nothing
* Lock VM from batch (batch id) -> VM handle
* Batch id: the id of the batch of VMs from which a VM should be locked
* Rules:
  + Upon this call, generate a VM handle #H; add {#H, state: PENDING, vmid: null} to VM handle list; push (batch id, #H) to pending locks list; decrement batch size if batch size greater than 0
  + Whenever an item (batch id, #H) is in the pending locks list for which the there is a VM #X in the batch whose state is READY
    - Remove a VM #X from the batch; occupy VM #X, set vmid of #H (in the handle list) to #X and state to ASSIGNED
* Release VM (VM handle #H)
* VM handle: the handle of the VM given by the lock VM operation
* Rules:
  + Upon this call
    - VM Handle:If #H is in the handle list (with state ASSIGNED) with VM #X
      * Clean up VM #X; remove #H from handle list
    - If #H is in the pending locks list (i.e. in handle list with state PENDING) with batch id #B
      * Remove #H from handle list and pending locks list
  + Whenever there is a batch #B for which (size of batch #B + #pending locks for #B) is smaller than (#VMs in that batch + #occurances of #B in the preparation queue)
    - If the preparation queue contains an item of #B
      * Remove the item
    - Otherwise if there is a VM #X in the batch with state WAIT
      * Remove #X from the batch
    - Otherwise
      * Remove VM #X from the batch; clean up #X
  + Whenever there is a batch #B for which (size of batch #B + #pending locks for #B) is greater than (#VMs in that batch + #occurances of #B in the preparation queue)
    - Push #B to the preparation queue
* Cancel batch (batch id #B)
* Batch id: the id of the batch to cancel
* Rules:
  + Upon this call
    - Set batch #B’s size to 0; mark batch for deletion
* Get VM Info (VM handle #H)
* VM handle #H: the handle of the VM given by the lock VM operation
* Rules:
  + Upon this call
    - If #H is in the handle list with state PENDING
      * Return PENDING
    - If #H is in the handle list with state ASSIGNED with VM #X
      * Return {state: ASSIGNED, vm: {data for VM #x}}

Other Rules

* VM states, state versioning, refresh / change times, ping, and timeout
  + The VM’s state is one of {CREATING, BOOTING, FREE, READY, OCCUPIED, ERROR, WAIT, KILLING}.
  + The VM’s state version is an integer which is incremented whenever the state is refreshed.
  + The refresh time is a timestamp that is reset to the current time whenever the state is refreshed.
  + The change time is a timestamp that is reset to the current time whenever the state is changed.
  + The ping interval P is a configured timespan; at any interval of P time if the VM has been in the FREE, READY, or OCCUPIED states, at least one ping must be executed.
  + A ping is replied by one of {OK, ERROR}. If OK is received, refresh the state; if ERROR is received, change the state to ERROR.
  + A VM’s initial state of CREATING and undergoes these transitions
    - When OpenStack replies with a booted VM info, transition CREATING -> BOOTING
    - When VM checks in, transition BOOTING -> FREE
    - When the prepare command is issued for the VM, FREE -> WAIT
      * If reply is OK, WAIT -> READY
      * Otherwise, WAIT -> ERROR
    - When the occupy command is issued for the VM, READY -> OCCUPIED
    - When the cleanup command is issued for the VM, READY/OCCUPIED -> WAIT
      * If reply is OK, WAIT -> FREE
      * Otherwise, WAIT -> ERROR
    - When the kill command is issued for the VM, [ANY] -> KILLING. At this point the VM is no longer considered part of the pool.
  + Whenever a VM’s refresh time is older than refresh timeout for the state (different for each state, but 0 for ERROR)
    - Kill the VM
* Batch cleanup
  + Whenever a batch is marked for deletion, has size 0 and has no VMs, and that the batch does not exist in any pending lock item, remove the batch