OPENWHISK TECH INTERCHANGE

CONCURRENT ACTIVATIONS

MOTIVATION

- Primary: Throughput improvement for steady-use actions
- Secondary: Fewer containers to operate (manage larger load with fewer containers, IFF there is steady traffic for at least some of those actions)

STEPS TO ACHIEVING CONCURRENCY

- Action image changes:
  - Disable inherent state/concurrency tracking in action image
  - Only nodejs (https://github.com/apache/incubator-openwhisk-runtime-nodejs/pull/41)
  - Interleaved logs - either prevent interleaving by buffering in nodejs, or force structure to log to include activation id (relies on queries via LogStore impl). For now:
    - bring-your-own-action-images
    - OR disable log collection
- Invoker changes
  - MessageFeed.maximumHandlerCapacity (peeking behavior)
  - ContainerPool.maxConcurrent (free -> busy behavior)
  - ContainerProxy.activeActivationCount (stay in Running state till all complete)
  - HttpUtils.maxConcurrent (use PoolingHttpClientConnectionManager)

QUEUEING BEHAVIOR

- Worst case: all activations are unique actions
  - Result: extra messages peeked and held in memory while all containers are busy for that invoker
  - Risk: in event of a crash, the number of lost messages will be greater than before (previously: <maximumContainers> messages lost, now: <maximumContainers*maxConcurrent> messages lost)
- Best case: all activations are same action
  - Result: all messages are processed on a single warm container
  - Risk: same risk for peeked message loss on crash, but less waiting will result in messages lingering in memory for a shorter period
- May reduce message peeking to <maximumContainers*maxConcurrent*concurrentPeekRatio> to adjust peek size to be between worst case and best case.

TESTS

- throughput.sh (existing)
- throughput-async.sh (new)
  - 175ms response delay
  - Emulates downstream API waiting (or other wait scenarios)
- Async (throughput-async.sh) example results (100 connections):
  - 114 RPS (maxConcurrent=200) vs 20 RPS (maxConcurrent=1) vs 5 RPS (master)
- Sync (throughput.sh) example results (100 connections):
  - 60 RPS (maxConcurrent=200) vs 60 RPS (maxConcurrent=1) vs 48 RPS (master)

DRAWBACKS

- Potential for activation state leaks
  - **Make sure the actions (and runtime images) do not introduce or rely on state that exists across activations (don't use globals, etc)**
- Interleaved logs
  - **Use a customized action image to buffers the activation logs internally (to prevent the interleave)**
  - **Use a LogStore impl to deal with log storage + fetching based on activation id**
    - **And… use a customized action image to structure logs to include the activation id in each log line**

NEXT STEPS

- Allow action devs to signal their own concurrency limits
  - Default = 1 (existing behavior unchanged)
  - Annotation: wsk action create … --annotation max-concurrent 200
  - Leave existing config (whisk.container-pool.max-concurrent) in place as a systemwide max
- Action images with structured (or buffered?) log options to use concurrency locally

FUTURE: MORE INTELLIGENCE

- Separate topic for concurrent actions
  - Need to manage consuming multiple topics to avoid starving non-concurrent actions
- Dedicated invokers
  - Allow some invokers to be reserved for concurrent actions
  - Because concurrent actions (may) have different traffic patterns and container lifecycle. (but what if they don’t actually get used concurrently?)
- Separate grace period to stop concurrent action containers