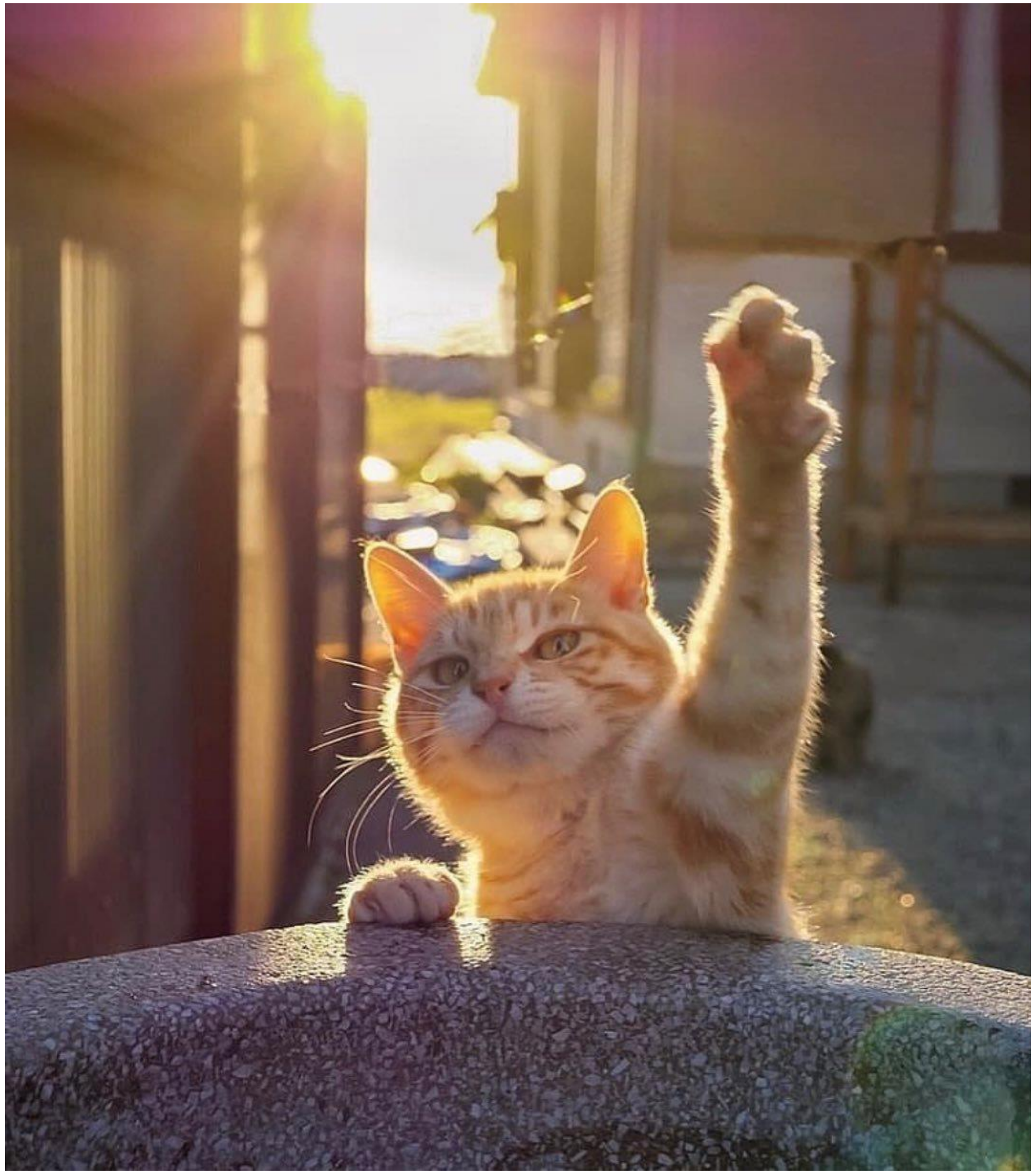
What’s new in 3.0 GC

Maoni Stephens

09/18/2019

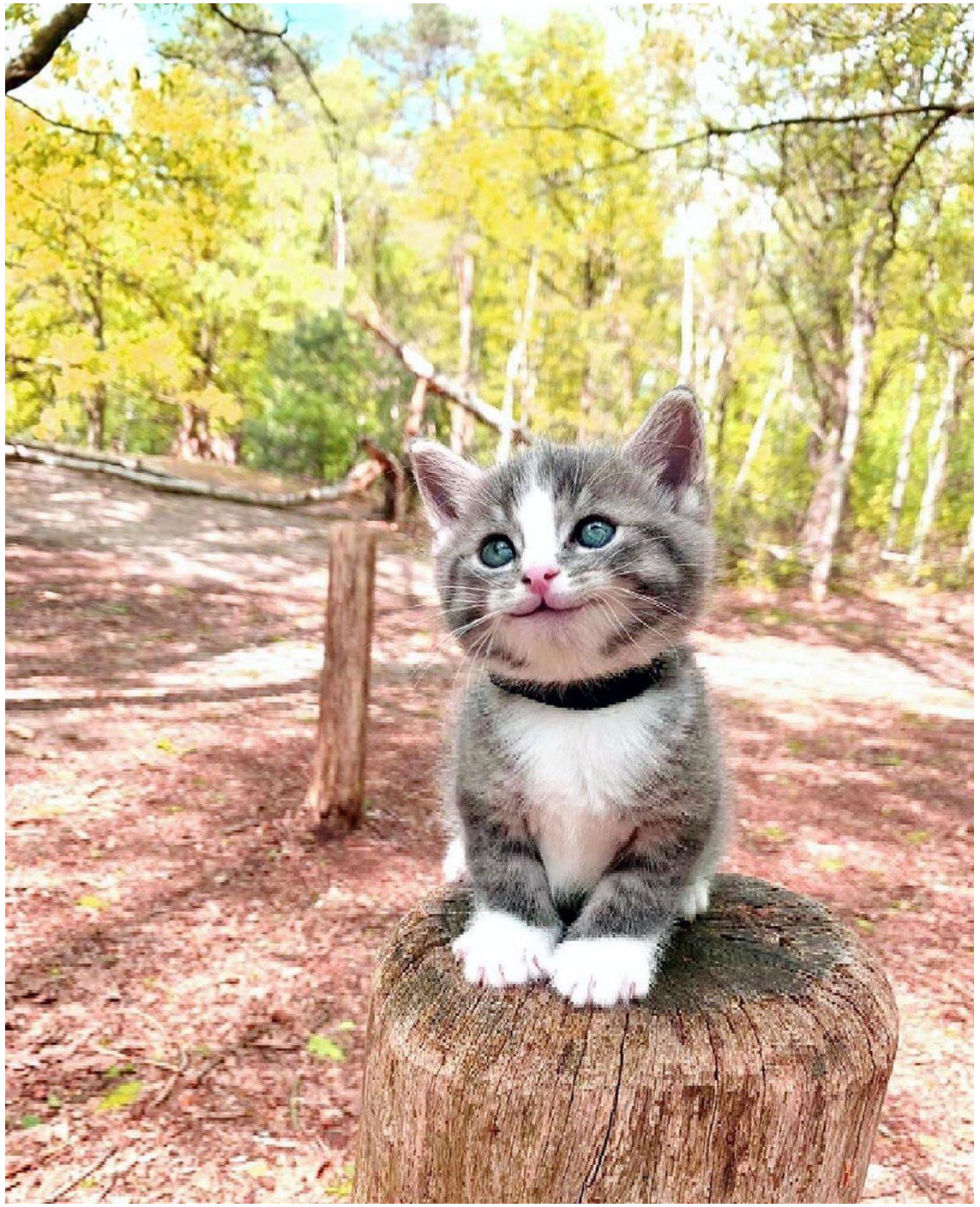
Hello!



Desktop port

- SOH/LOH alloc lock split
- A new provisional mode
- 3 new configs
  - GCHeapAffinitizeMask, GCHighMemPercent, GCLOHThreshold
- Others

Wanna know some implementation details?



How does locking work on the GC heap during allocation?

- First thing first, when would allocation actually come to the GC?
- SOH and LOH shared the same lock for the same heap
- LOH free list usage
  - LOH sweeping threads the LOH free list
  - LOH allocation consumes LOH free list

PerfView

"LOH Allocation Pause (due to background GC) > 200 Msec for Process X" table

"Waiting for BGC to thread free lists"

How to fix this

- Observation – sweeping LOH usually takes much less than sweeping SOH
- By splitting the SOH and LOH alloc lock, we can have BGC LOH sweep take the LOH alloc lock
- Reduced Bing Backend P95 latency by 10%
- Better ways to fix this but more complex

High memory load situation

- Physical memory load >= 90%
- Can be changed with the GCHighMemPercent config
- New provisional mode says we will do a gen2 GC when we observe that gen1 GC increases gen2 size

GCLOHThreshold

- You can only make it bigger
- How you should use this config
- New profiling API ICorProfilerInfo10::GetLOHObjectSizeThreshold
- New Profiling mask COR_PRF_MONITOR_LARGEOBJECT_ALLOCATED for large object allocations

Container scenarios in 2.2-

Container scenarios – previous problems

- A lot of the checking/decision was happening during a GC, examples
  - Check for “Is doing a gen2 going to be productive”
  - Heap expansion did not take commit into consideration (only reserve)
  - When we have multiple segments on a heap, now the tuning sees the new ephemeral seg which is not at all full
  - Logic to downgrade a gen2 GC to a gen1 GC
- Check for “Are we short on seg when we just come out of a GC”
  - Based on min budget which is based on LLC size

Container scenarios – 3.0 implementation

- Disallow heap expansion (reserve up front)
- Check for limit during commit code paths in allocator
- Hardened tuning parameters against limit, eg
  - Segment size
  - Allocation budget
- Compact LOH if needed
- Track GC's own bookkeeping against limit
- Retry for Server GC in allocator

Choosing alloc heap in Server GC allocator

- We find a heap that's relatively empty
- But also try to keep on the current core for a while
- NUMA considerations
- The choosing part is done without a lock
    - Which means when we actually allocate, some other thread could have beat us to it
    - We need to retry if we are not to the limit
    - New alloc state introduced

Large pages

- Need to set hardlimit config too (unless it’s already set by the virtue of having a memory limit on a container)!
- Current state (for GC) –
  - GC heap can be on large pages
  - GC’s own bookkeeping is not
- More support coming

Future focus

- Continuing latency reduction
- Allowing users to communicate to us which perf aspects are most important for them
  - Memory footprint
  - % CPU in GC
  - Individual GC latency

Q&A