ML for Memory Allocation

Jan 22 2026

Problem setting

- Memory allocators (e.g., glibc, tcmalloc, jemalloc, Hoard)
  - Performance vs memory consumption
  - Optimizations: per-thread buffers, arenas
  - Object-size classes to avoid bloat

- Allocation lifetime also important
  - Long lived objects cause hugepage fragmentation

- C++ vs managed runtimes (garbage collection)

Challenge

- How to predict allocation lifetime?

- Naïve approach:
  - Profile an application – record lifetimes at each allocation site
  - Drawback: not generalizable (revisions, variations across executions)

- Online approach:
  - Overhead
  - Unseen contexts

LLAMA contributions

- ML to predict lifetime using context

- Challenges:
  - Training? How to generalize?
  - Prediction overheads
  - Mispredictions

- Follow-up work: TelaMalloc allocation for ML accelerators

Discussion

1. When is ML a “viable” tool for memory allocator? What signals make the problem “learnable” here? When it make it a bad fit?

2. The design uses many constants (e.g., K, M, lifetime classes, region size) . Are there any benefits to tuning them? Is yes, how would the values affect performance and fragmentation?

Title: [Lecture 4] Discussion