# PRmalloc: Leveraging Predictability for Deep Learning Memory Allocation

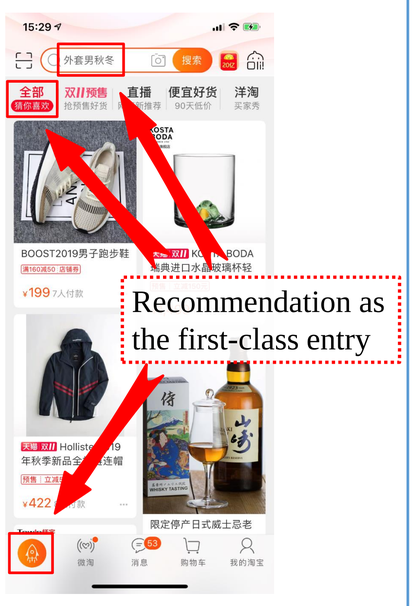
Wencong Xiao, Shiru Ren, Tongxuan Liu, Yong Li

PAI Team, Computing Platform, Alibaba Group

## Deep Learning Recommendation in Alibaba

### Industrial-scale recommendation

- Train models with **massive sparse** user behavior data on **CPU**
- Consume a large amount of **host memory**
- Scale to **thousands** of servers on TensorFlow
- Occupy more than **50%** memory compared with the real requirement
- Up to **900K/s** minor page fault during job execution



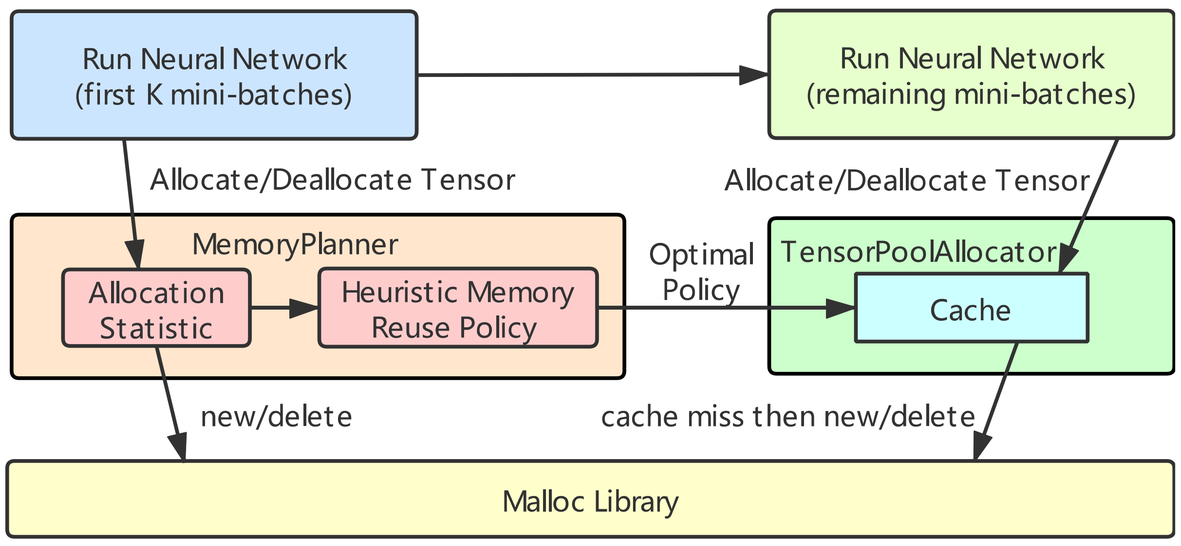

### Existing memory allocators: A poor fit to deep learning

- Those memory libraries prominently improve the performance for web server, benefiting from a per-thread memory pool design for **small memory blocks** (e.g., < 32KB).
- The tensors used in training usually require **large memory blocks** (~MB), and the summary of total required memory can be tens of GB.

### The predictability of deep learning training

- Includes millions of **mini-batches**, each mini-batch is a traversal on a deterministic data flow graph for computation.
- Most of the allocate/deallocate requests are **consistent** among mini-batches.
- The **dependency relationship** of computation can also be utilized to schedule memory block recycle.

### PRmalloc: Predictable Reusable memory allocator




### Design overview

We introduce PRmalloc, a brand new *predictable reusable memory allocator* taking advantage of such remarkable predictability. As indicated in the figure, in contrast to conventional memory allocators, PRmalloc adapts to the memory usage characteristics of different deep learning training applications, by learning from the memory block life-cycle at the early stages through *MemoryPlanner* module. The collected domain knowledge helps to generate heuristic policy into *TensorPoolAllocator* to better schedule the memory recycle.

### Two vital benefits

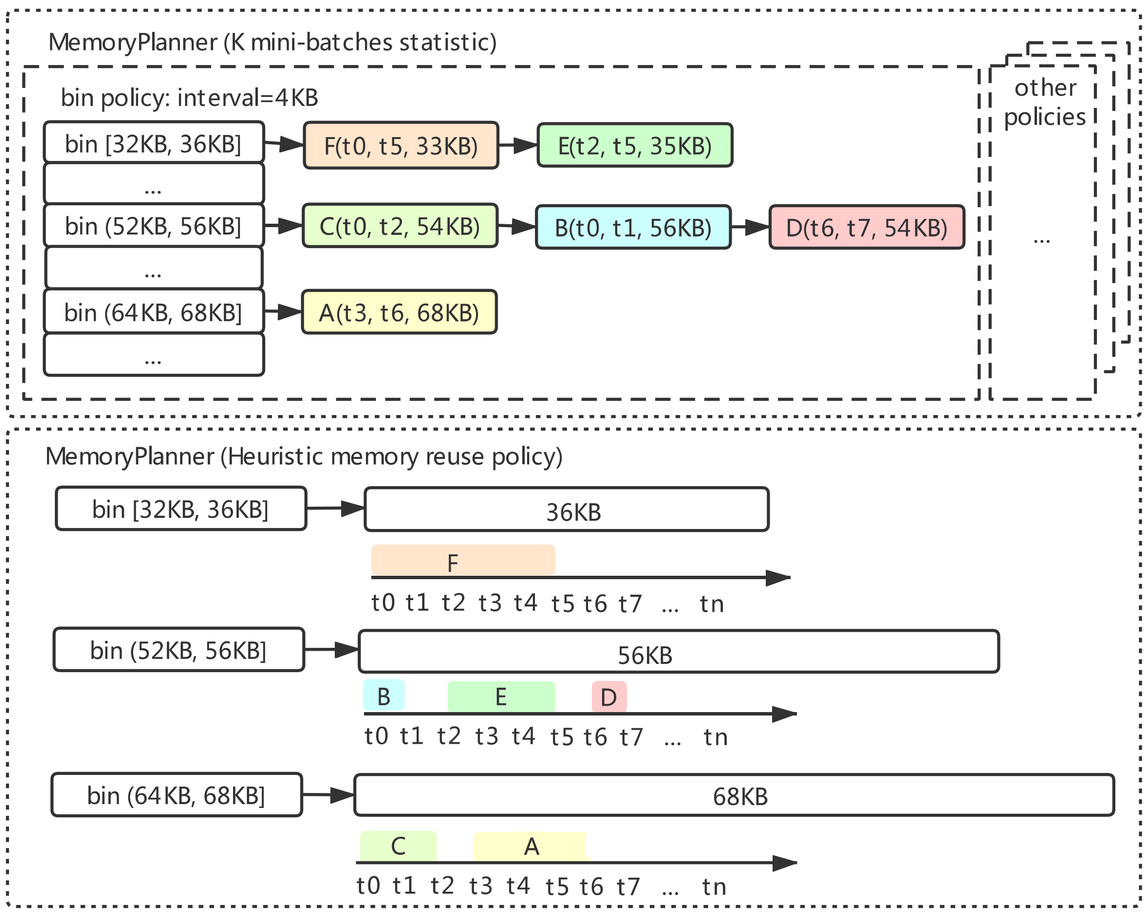
- Memory allocator can cache the large memory block, saving the minor page fault introduced by memory allocation system call.
- A better memory reuse plan can be learned to reduce the memory footprint, minimizing the overall resource usage.

## System architecture and workflow

*MemoryPlanner* first collects the tensor allocation information during execution. Based on this information, *MemoryPlanner* can heuristically search for the optimal memory allocation scheme for the running deep learning application. By leveraging the malloc library, *TensorPoolAllocator* uses a lock-free queue to manage the memory blocks' allocation and deallocation according to the optimal memory allocation scheme.

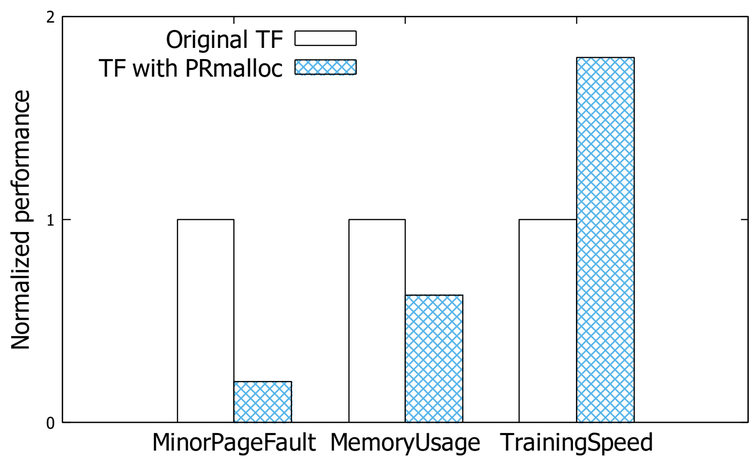
Thanks to the predictability, we observe that the read/write sequences of the tensors remain quite stable between successive mini-batches. To find the optimal memory allocation scheme, the *MemoryPlanner* first leverages the allocation statistic to record all tensors‘ allocate and deallocate operations in the first K mini-batches. Based on the estimations, it heuristically searches for the optimal bin policy to minimize the memory usage and minor page fault. *TensorPoolAllocator* considers the identified heuristic policy to manage a memory pool for the rest of mini-batches.

### Heuristic memory reuse within mini-batches




To further cut down the memory footprint, PRmalloc also reuses memory blocks by utilizing the memory dependency information within a mini-batch. During allocation statistics recording, PRmalloc first sorts all memory allocation requests within a mini-batch from largest to smallest according to the size of the request. Then, PRmalloc tries to match each request with the appropriate-sized block (which is available for the requested time period) by ascending order of the bin size.

### Evaluation




Deep-CTR Performance

We use a real production workload from Taobao Search of 210 Alibaba to evaluate PRmalloc. Experiment result shows that PRmalloc boosts the training speed of state-of-art recommendation models by 1.8×, benefiting from lower memory footprint and fewer page fault.