Virtual Address Translation via Learned Page Table Indexes

Artemiy Margaritovy
Dmitrii Ustiugovz
Edouard Bugnionz
Boris Groty
Learned Address Translation Model

• Previous work:

• A learned model could effectively replace a B+Tree for indexing a sorted key range.
Weakness in directly learned model

• High inference time in software-based learned models
• It is hard to learn the distribution of randomly scattered physical address
• If the prediction is wrong, memory leak will happen
Proposed method: a more pragmatic strategy

- Predicted PTE
- Tag Check
  - False (Wrong Prediction)
  - Last level radix page table
- Physical Address
- Conventional Page Table
Proposed method: to accelerate speed

• Reduce complexity

• Only the page address needs to be predicted (PTE = \texttt{page addr} + offset)

• Lowing PTE location prediction accuracy: produce a range of possible locations (multiple PTEs could be fetched in parallel, and memory bandwidth is sufficient)
Proposed method: to integrate into modern systems

• Pre-defined NN structure. Only weights are determined at application time
• Training time could be amortized over the long lifetime of application
• Training can occur as a background task when system is idle
• Conventional radix page table could be used before NN is trained
Quantifying Analysis: Radix Tree

• Radix tree: optimized for low latency traversals. Used in memory management in Linux kernel
Quantifying Analysis: Software-based learned indexes

• 3-level page table
• Two-level hierarchy of models: 1 in the first level, and 32 in the second level
• The third level is a radix tree page table

• Each model: Three layers NN (27 – 32 – 1)
• Accuracy: 99.9%
Quantifying Analysis: Software-based learned indexes

- Not capable for the need of lower-latency learned Index architecture

![CPU cycles to produce target PTE location](chart)

- Radix page table
- Learned model
- Learned model with reduced size
Future directions:

• Reduced precision -> **reduced complexity**
• Using a microarchitectural learned page table indexer
• Binarization of weights and activations -> replace complex multiplication with simpler boolean operators