



How expensive is it? Big data for ML cost modeling



UC DAVIS

COLLEGE of ENGINEERING

College of Engineering
University of California, Davis



HOW EXPENSIVE IS IT? BIG DATA FOR ML COST MODELING

Presented by Aiden Grossman

April 28, 2024

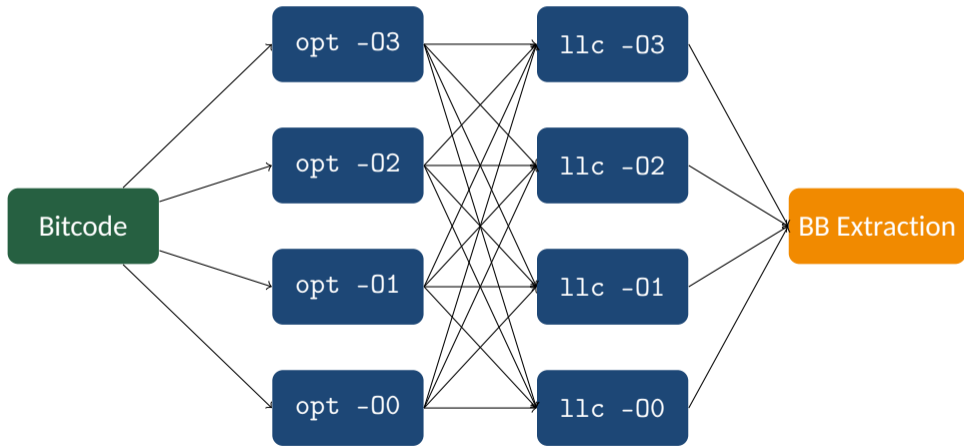
Work performed while employed by LLNL.

WHY STATIC LEARNED COST MODELS?

- » Benchmarking is expensive, noisy, and can require restricted hardware features (eg perf counters).
- » Deterministic performance results can make training ML models easier.
- » When appropriate benchmarking tooling is available, bringing up new architectures is trivial. This is not the case for analytical cost models which require a significant amount of reverse engineering.

- » We construct a dataset of about 1.5B X86 basic blocks using ComPile.
- » We use the BBAddrMap tooling to extract individual basic blocks.
- » We postprocess the extracted blocks to remove call, return, branch, and syscall instructions.
- » For initial results, we filter instructions that may load or store to simplify validation of the benchmarking infrastructure.

DATASET CONSTRUCTION



- » We need ground truth values to build a learned cost model.
- » We collect inverse throughput values using `llvm-exegesis`.
- » We use some custom python scripts to parallelize the collection. Results aren't exactly deterministic (most likely due to variance in `ioctl` syscalls), but can be within about 0.2% run-to-run variance depending on the system.

- » Benchmark data does not need to be perfectly accurate for training!

Configuration	Run-to-run Variability
1 benchmark + no HT	~ 0.1%
1 benchmark/core + no HT	~ 0.5%
1 benchmark/thread + no HT	~ 0.6%
1 benchmark	~ 0.8%
1 benchmark/core	~ 1.2%
1 benchmark/thread	~ 5%

- » Basic blocks that load or store to memory will segfault if the memory isn't mapped into the virtual address space.
- » We currently use an iterative technique where we map at the address of a segmentation fault iteratively until we no longer segfault or hit an upper limit on the number of mapped pages.
- » This is done using the memory annotations feature in `llvm-exegesis`.

DEALING WITH MEMORY ACCESSES

Before annotating:

```
movq $8192, %rax
movq (%rax), %rdi
```


After annotating:

```
# LLVM-EXEGESIS-MEM-DEF mem1 4096 7fffffff
# LLVM-EXEGESIS-MEM-MAP mem1 8192
movq $8192, %rax
movq (%rax), %rdi
```

- » Our raw deduplicated dataset is about 1.5B basic blocks.
- » The dataset after processing is about 1B basic blocks.
- » The dataset post-processing along with removing instructions that access memory produces about 150M basic blocks.
- » This is a significant improvement over previous open datasets (about 300K BBs) and closed datasets (about 1.5M BBs).

- » We build off the pre-existing GRANITE architecture with some slight differences in the training.
- » We use a cosine learning rate schedule and use similar hyperparameters to those presented in the original paper.
- » Previous experiments showed significantly improved performance going from smaller datasets (about 300K BBs) to larger datasets (about 1.5M BBs). We take this to the extreme.

INITIAL RESULTS



BB Count	MAPE
1M	14.3%
2.5M	5.5%
10M	5.8%
10M ¹	4.7%

- » These results are for znver2.
- » Previous results showed MAPEs on the order of 6-8% for larger datasets (1M BBs) and 8-10% for smaller datasets (300k BBs).

¹With adjusted hyperparameters

WORK TO BE DONE

- » Training on the full dataset.
- » Validation and training on datasets including memory-accessing instructions.
- » Hyperparameter tuning.

- » Fine-tuning of these models with additional context, such as cache misses. (See Viraj Shah's presentation).
- » Beyond basic-block level cost modelling using similarly large datasets, potentially taking advantage of input generation.
- » Training MLRegalloc models using this new cost modelling technique and evaluating performance.
- » Mutation-based fuzzing to make the model invariant against changes such as register permutations.

- » The tooling is productionized. The vast majority of the benchmarking tooling is available in upstream LLVM, is decently tested, and reasonably reliable at this point.
- » Better model performance for future work, much more comparable to the most accurate static cost models.

ACKNOWLEDGEMENTS

The presenter would like to thank the following people for their guidance/code reviews:

- » Ondrej Sykora
- » Mircea Trofin
- » Clement Courbet
- » Guillaume Chatelet
- » Johannes Doerfert

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-PRES-862639).

QUESTIONS?



Answers! (Hopefully)