Modern Compiler Optimization

WAX SRCPR

Fabrice Rastello. Inria CORSE
A compiler transforms a program into a binary.
A combinatorial number of possible candidates exists.
Each showing different performance (speed, energy consumption).
The objective of compiler optimization is to generate/choose the “best” one.
The compiler needs a performance model and a search strategy.
Classical Compiler Framework

- The classical framework consists of several consecutive phases that iteratively transforms the intermediate representation (IR).
- They use metrics to drive heuristics.
- With regard to the overall search space, each phase makes partial choices.
Metric vs Performance

- Even with “good” metric, early decision without any knowledge of further search space may lose opportunities.
- For example, maximizing the operational intensity might constrain too much other choices (e.g. parallelism).

Operational intensity:

\[ OI = \frac{\#\text{Computed_Flops}}{\#\text{Communicated_Bytes}} \]
Make Multiple Random Choices

Take several candidates (instead of one) and use random draws instead of deterministic total order.
A model is only as good as it can.

But it does not have to perfectly predict actual performance.

Its quality should be measured as its ability to reinforce the probability to select good candidates.
Modern compiler optimization needs

- Randomized heuristics to take partial decisions
- To measure/express the quality of used metrics and adapt the search strategy consequently (expansion vs exploitation)
- Metrics/models fast to evaluate
- Compiler infrastructure that handles very efficiently transformations as transactions
Short Quiz

Which Conv2D binary is faster between the one from?

- A generic compiler (e.g. icc, gcc, LLVM)
- The manufacturer optimized library (e.g. oneDNN)
- A domain specific compiler (e.g. TVM Ansor)

How much faster?