High Performance Computing Systems (CMSC714)

Lecture 14: Autotuning

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Summary of last lecture

- Isoefficiency

$$t_o = K \times t_1$$

- Helps us understand scalability and computation-communication tradeoffs
- Performance modeling
- Analytical: LogP, alpha-beta model

Autotuning

- Ultimate goal: performance portability — reasonable performance as we move from one architecture to the next
- Generation and exploration of a search space to identify the best performing option
  - Evaluated through models or empirical measurement
- Search space:
  - Code variants
  - Application parameters
  - System parameters

Different approaches

- Empirical autotuning
  - Execute each code variant or parameter combinations to identify the best performing one
  - Can also use runtime prediction models instead of running code
- Code variants
  - Code organization, data structures, algorithms
  - Parallelization strategies
  - Data movement optimization: data placement, blocking/tiling

Exploring the search space

- Brute force: try every option in the search space empirically
- How to limit the search space to a subset?
- Model-free: simulated annealing, genetic algorithms
- Model-based: analytical/empirical/machine learning models
  - Limited by accuracy of models

Software Engineering Challenges

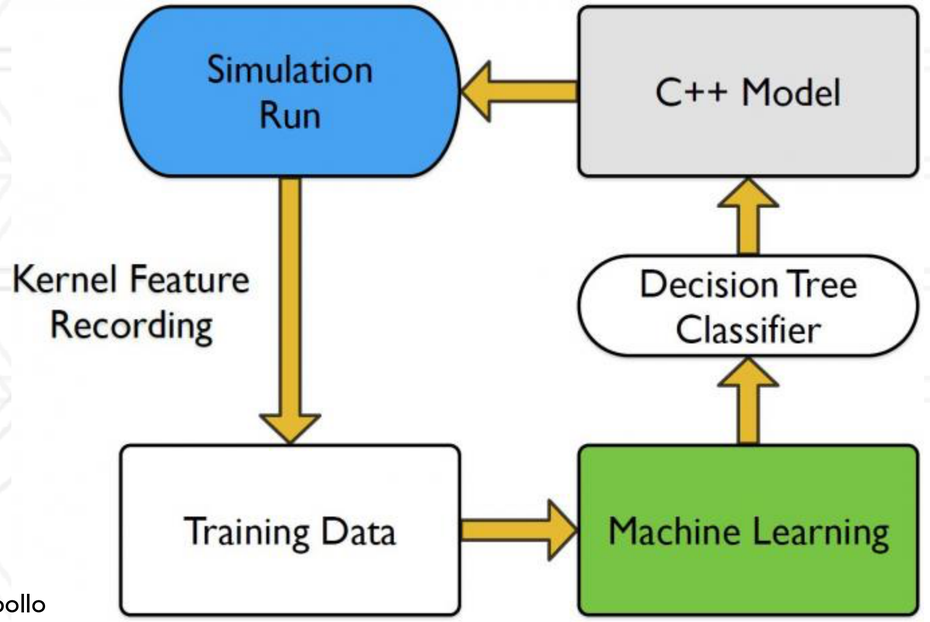
- Offline auto-tuning can make compilation slow
  - Many variants need to be executed
- Empirical auto-tuning involves the developer in the process
- Build process for auto-tuned code can be complex
- Debugging auto-tuned code can be challenging

Libraries

- Isolate performance-critical sections behind a standard API
- ATLAS, Spiral, FFTW
- Automatically Tuned Linear Algebra Software based on Automated Empirical Optimization of Software (AEOS)
- Goal: Portable efficient implementation of BLAS
  - Blocking factor, different source code implementations
- Goal: Generate an L1 cache-contained matrix multiply kernel

Application-level Tools

- Tools allow expressing tunable parameters and exposing code variants
- If performance depends on input, tuning must be done at runtime
  - Active Harmony, UMD
  - Apollo, LLNL



https://computing.llnl.gov/projects/apollo

Questions?

UNIVERSITY OF
MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu