Machine Learning for Systems

A peek for researchers

Benoit Rostykus

December 6th, 2020
NeurIPS Netflix presentations

NETFLIX

ML **for** systems

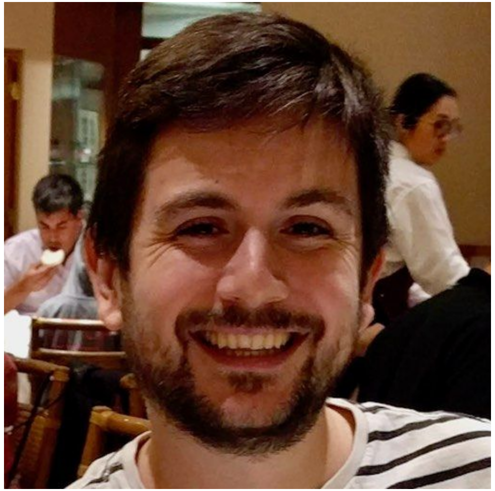
Your host today

- Senior Machine Learning Researcher
- Current focus: make Netflix container platform smarter

- In the past:
  - retention modeling, causality
  - large-scale convex optimization
  - Low-latency computation

- Love production systems & constrained environments

"Necessity is the mother of invention"



Teaser: did you know?

A perceptron has likely been hiding deep inside your CPU for a while

BPU: Branch Prediction Unit

- Silicon predicting which branch will execute in an “if” statement
- Very important
  - for performance, CPUs do speculative execution
  - instructions execution is pipelined
- Typical misprediction rate: 1-10 mispredictions per 1k instructions
- Typical misprediction cost: 20 cycles

Some advanced branch predictors have “mini linear models” printed in silicon

(Confirmed in AMD Zen and Samsung M1 microarchitectures)

Agenda

- Why you should care
- Selected Netflix problems
  - Noisy neighbors
  - Oversubscription
- A wide space

Why you should care

Most of the systems you interact with as a researcher are heuristics-driven:

- Operating System
- Compiler
- Garbage collector
- Cloud scheduler (Spark...)
- Database

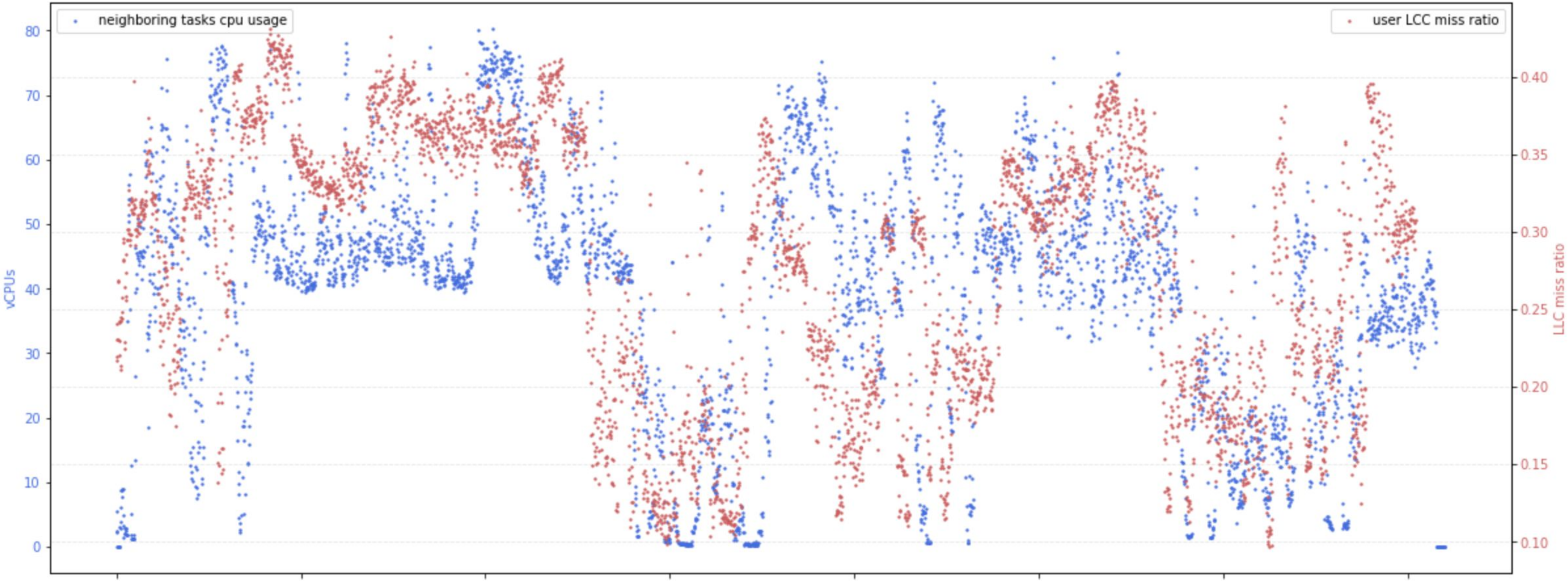
ML is good at automating heuristics with data
Relatively new field when applied to systems

Selected Problem: noisy neighbors

- Levels of caches (L1, L2, LLC…) are usually shared across CPUs
- Creates interference: more or less cache misses depending on access patterns
- Degrades software performance in colocated environments
- Linux task scheduler (CFS) suboptimal

Problem is combinatorial, stateful, dynamic

Selected Problem: noisy neighbors



from automatic second-granularity Performance Monitoring Counters data collection

Selected Problem: noisy neighbors

Implemented a user-space solution based on combinatorial optimization

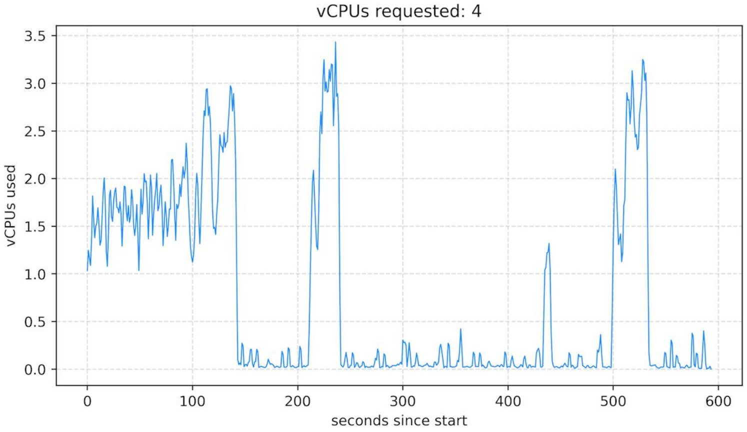
- Every ~10-60s, each host runs a **M**ixed **I**nteger **P**rogram to map its running containers to its CPUs
- Goal: minimize cache misses

AB test results:

- Improved tail latency of critical Netflix services by 30%
- Decreased long-running batch jobs outliers

Selected Problem: oversubscription

Facts:
- Cloud resources are underutilized
- Manual right-sizing doesn't scale
  - resource utilization is variable over time
  - developers should focus on app logic
  - developers ask for more than what they need
- VM auto scaling is slow/not granular enough

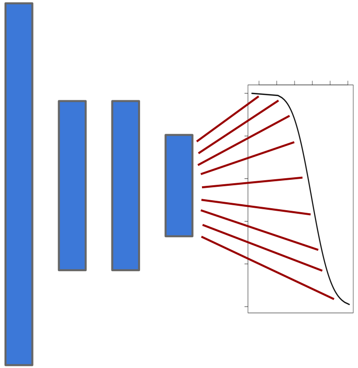


Opportunity: decide when and where to run containers

Goal: learn software execution behavior to automate scheduling, sizing and placement

Selected Problem: oversubscription

Some predictive tasks:
- Runtime duration of batch jobs
- Cpu, memory, disk, network usage of containers



Techniques and challenges:
- Non-Gaussian behaviors
  - Underestimating is worse than overestimating
  - Outliers drive perceived performance
- Model distribution tails through large-scale conditional quantiles regression
- Time series prediction
- Delayed scheduling: when to launch?
  - Model Predictive Control

Predictive models integrated inside platforms such as Kubernetes to drive scheduling decisions

A wide space

There’s an ∞ number of problems to solve
and an ε number of people on it

here is a non-exhaustive selection of interesting ones….

Compilation

Profile Guided Optimization (PGO)

1) Instrument code execution to collect tracing data
2) Run code
3) Re-compile, leverage traces to make better heuristics decisions

=> Basic “learn to compile” data loop

Some examples:

- RL-driven inliner in LLVM
- AutoFDO
- Windows reports 5-20% performance improvements
- Chrome reports 10% faster page load

Heterogeneity creates opportunities

Moore's Law stopped around 2000-2005

Free lunch is over.

- 2000-2015: increase parallelism (multi-cores)
- 2015-now: increase specialization

Both approaches:

- require more work from software engineers
- can benefit from ML

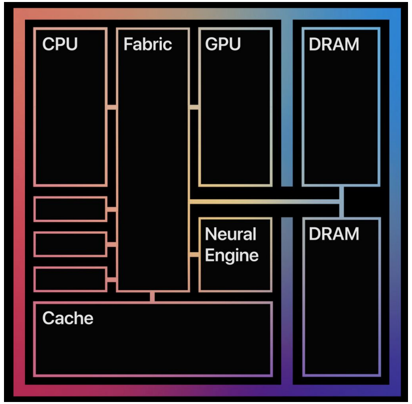
"The only path left to improve energy-performance-cost is specialization. Future microprocessors will include several domain-specific cores that perform only one class of computations well, but they do so remarkably better than general-purpose cores"

Computer Architecture: A Quantitative Approach (6th ed.) - JL Hennessy, DA Patterson

Heterogeneity creates opportunities

On chip

example: Apple M1 SoC (Nov. 2020)



- 4 “high perf” cores
- 4 “high efficiency” cores
- 8-core GPU
- 16-core Neural Engine
- Unified Memory

Some problems:

- Scheduling across heterogeneous units
  - Compile time? JIT? On-chip controller?
- Caching algorithms
- Optimize for:
  - Throughput?
  - Latency?
  - Power efficiency?
  - Schedulability?

Heterogeneity creates opportunities

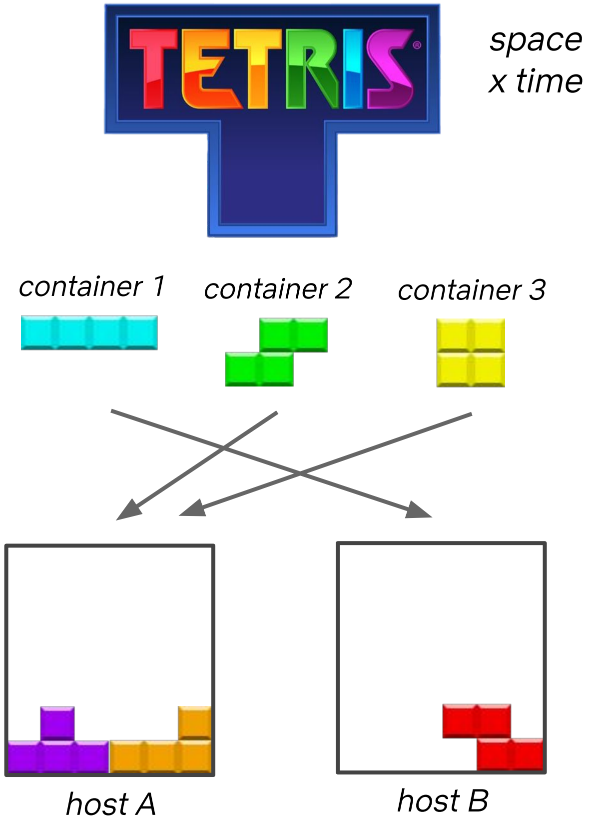
In the datacenter

- Complex dependency graph between:
  - services
  - data
  - batch jobs
- Multitude of constraints per application
  - SLA, hardware needs...
- Infinite service offering from cloud providers
  - 315 aws ec2 instance types as of Nov 11

Scheduling:

- Minimize cloud bill?
- Minimize time to completion?
- Maximize “performance”?
- Minimize hardware availability risk?

“Learn to schedule”



Memory: a leaky abstraction

Memory is a leaky concept

- Cache hierarchy
- NUMA domains
- Memory ordering
- Accelerators (such as GPU) memory
- DMA and RDMA

Programs, operating systems and distributed systems make assumptions on how memory is accessed.

Idea: learn memory access. Already applied to:

- prefetching
- allocators
- data structures
- ...

Operating System

Trend: more powerful kernel APIs for user-space extensibility/control
- cgroups v2
- eBPF

Doing more user-space allows for ML-driven approaches to:
- replace kernel heuristics
- Auto-tune the OS at runtime for a given application

Cooperative multitasking (userland, Goroutines) vs preemptive multitasking (threads)

Parting thought

Data-driven compilation, planning & execution of software will increase in importance

ML will be at the center of it

Thank you!

NETFLIX

Bibliography

- Erven Rohou, Bharath Narasimha Swamy, André Seznec. *Branch Prediction and the Performance of Interpreters - Don't Trust Folklore*. International Symposium on Code Generation and Optimization, Feb 2015, Burlingame, United States. Ffhal-01100647f
- Troffin M. *RFC: a practical mechanism for applying Machine Learning for optimization policies in LLVM*. LLVM mailing list, April 2020
- Haj-Ali, A., Ahmed, N.K., Willke, T., Shao, S., Asanovic, K. and Stoica, I., 2019, December. *Learning to vectorize using deep reinforcement learning*. In Workshop on ML for Systems at NeurIPS.
- Hashemi, M., Swersky, K., Smith, J.A., Ayers, G., Litz, H., Chang, J., Kozyrakis, C. and Ranganathan, P., 2018. *Learning memory access patterns*. arXiv preprint arXiv:1803.02329.
- Chen, D., Moseley, T. and Li, D.X., 2016, March. *AutoFDO: Automatic feedback-directed optimization for warehouse-scale applications*. In 2016 IEEE/ACM International Symposium on Code Generation and Optimization (CGO) (pp. 12-23). IEEE.
- Bearman, I. *Exploring Profile Guided Optimization of the Linux Kernel*. Linux Plumbers Conference, 2020
- Christoff M. *Chrome just got faster with Profile Guided Optimization*. Chromium Blog, 2020
- Kraska, T., Beutel, A., Chi, E.H., Dean, J. and Polyzotis, N., 2018, May. *The case for learned index structures*. In Proceedings of the 2018 International Conference on Management of Data (pp. 489-504).