Linux Kernel Autotuning

Cong Wang, Jasmine Mou
System Technologies & Engineering

ByteDance 字节跳动

1. Why Autotuning
2. Why Machine Learning
3. Machine Learning Approaches for Optimization
4. Statistical Performance Evaluation
5. Our Use Cases
6. Kernel Machine Learning

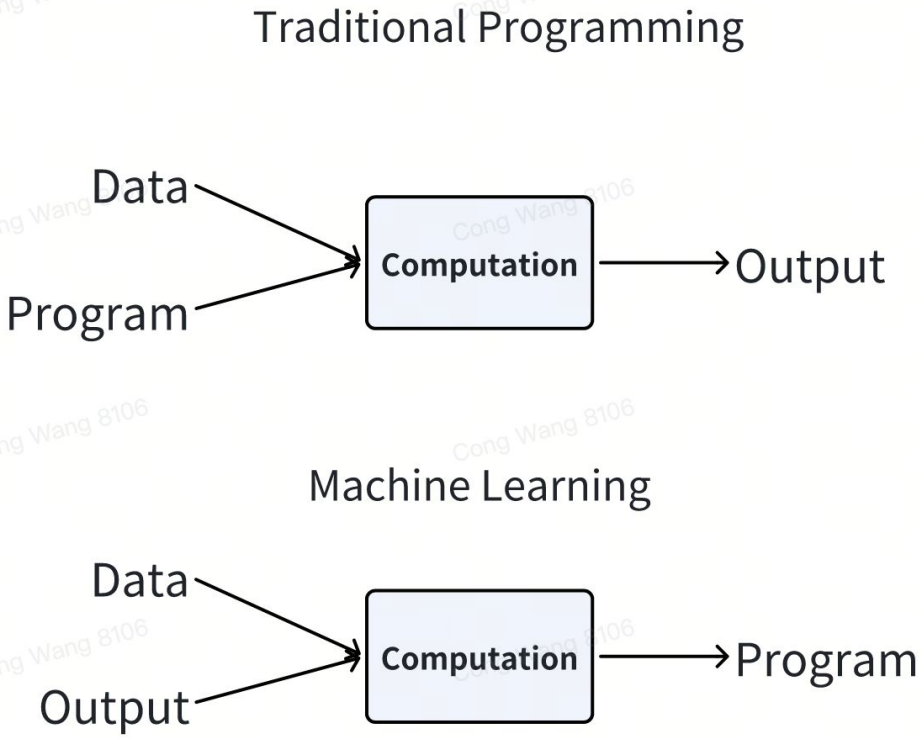
Disclaimer

- Please don’t ask me machine learning questions. 🤡
- I am trying to kill our jobs, just kidding…

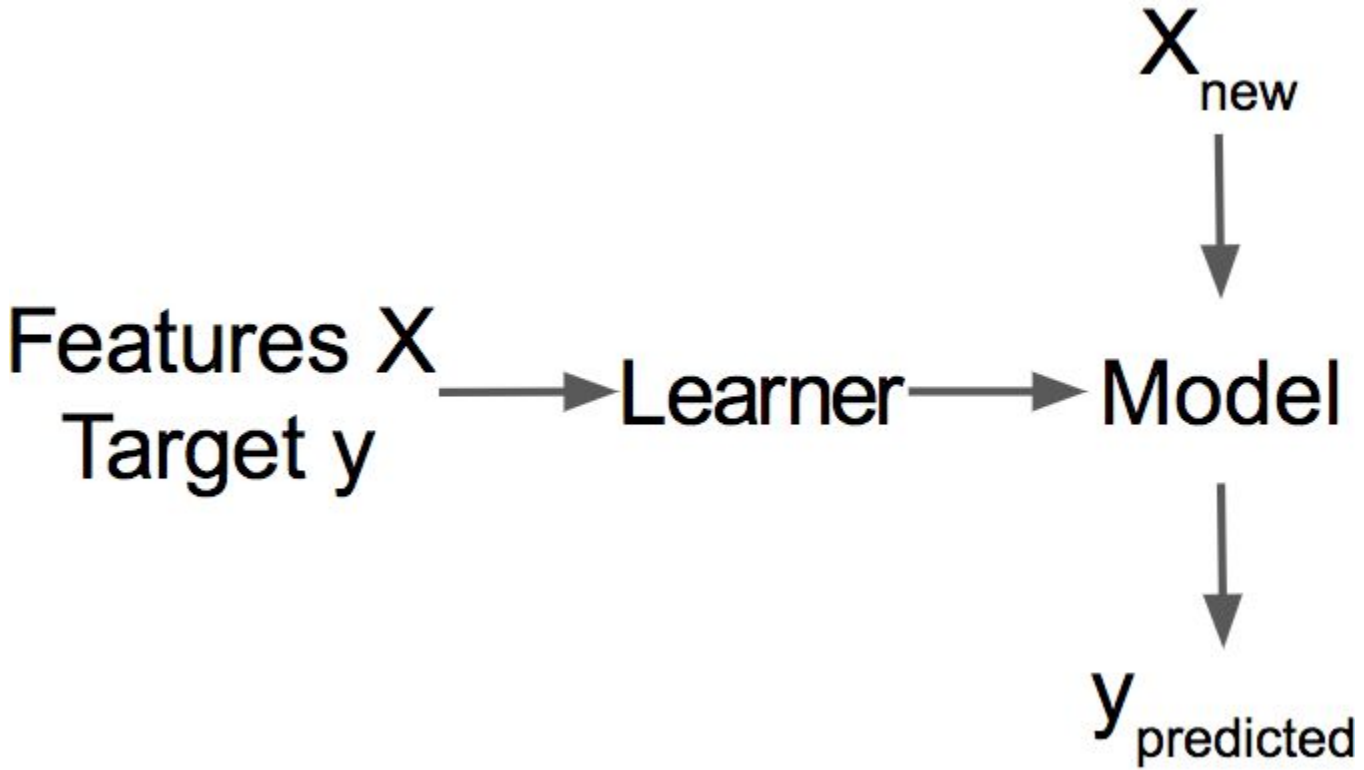
Why Autotuning?

- Liberate human engineers from tuning performance for each individual workload.
- Make better decisions with historical data when humans struggle.
- Find a more optimal solution than current heuristic method for a specific workload.
- How:
  - Rule based/heuristics
  - Machine learning based

Why Machine Learning?



Why Machine Learning?



Computer Systems are Filled With Heuristics

Compilers, Networking code, Operating Systems, …

Heuristics have to work well “in general case”

Generally don’t adapt to actual pattern of usage

Generally don’t take into account available context

Source: http://learningsys.org/nips17/assets/slides/dean-nips17.pdf

Machine Learning for OS

```
576             /*
577              * It's the expected callback index, assume sequential access.
578              * Ramp up sizes, and push forward the readahead window.
579              */
580             expected = round_up(ra->start + ra->size - ra->async_size,
581                                 1UL << order);
582             if (index == expected || index == (ra->start + ra->size)) {
583                     ra->start += ra->size;
584                     ra->size = get_next_ra_size(ra, max_pages);
585                     ra->async_size = ra->size;
586                     goto readit;
587             }
```

Source: https://github.com/torvalds/linux/blob/master/mm/readahead.c#L552

Machine Learning for OS

```
267             t = (s32)(tcp_jiffies32 - ca->epoch_start);
268             t += usecs_to_jiffies(ca->delay_min);
269             /* change the unit from HZ to bictcp_HZ */
270             t <<= BICTCP_HZ;
271             do_div(t, HZ);
272
273             if (t < ca->bic_K)              /* t - K */
274                     offs = ca->bic_K - t;
275             else
276                     offs = t - ca->bic_K;
277
278             /* c/rtt * (t-K)^3 */
279             delta = (cube_rtt_scale * offs * offs * offs) >> (10+3*BICTCP_HZ);
280             if (t < ca->bic_K)                            /* below origin*/
281                     bic_target = ca->bic_origin_point - delta;
282             else                                          /* above origin*/
283                     bic_target = ca->bic_origin_point + delta;
284
285             /* cubic function - calc bictcp_cnt*/
286             if (bic_target > cwnd) {
287                     ca->cnt = cwnd / (bic_target - cwnd);
288             } else {
289                     ca->cnt = 100 * cwnd;              /* very small increment*/
290             }
291
292             /*
293              * The initial growth of cubic function may be too conservative
294              * when the available bandwidth is still unknown.
295              */
296             if (ca->last_max_cwnd == 0 && ca->cnt > 20)
297                     ca->cnt = 20;   /* increase cwnd 5% per RTT */
```

Source: https://github.com/torvalds/linux/blob/master/net/ipv4/tcp_cubic.c#L214

Machine Learning for OS

```
3113            /*
3114             * If a workload spans multiple NUMA nodes, a shared fault that
3115             * occurs wholly within the set of nodes that the workload is
3116             * actively using should be counted as local. This allows the
3117             * scan rate to slow down when a workload has settled down.
3118             */
3119            ng = deref_curr_numa_group(p);
3120            if (!priv && !local && ng && ng->active_nodes > 1 &&
3121                                    numa_is_active_node(cpu_node, ng) &&
3122                                    numa_is_active_node(mem_node, ng))
3123                    local = 1;
3124
3125            /*
3126             * Retry to migrate task to preferred node periodically, in case it
3127             * previously failed, or the scheduler moved us.
3128             */
3129            if (time_after(jiffies, p->numa_migrate_retry)) {
3130                    task_numa_placement(p);
3131                    numa_migrate_preferred(p);
3132            }
```

Source: https://github.com/torvalds/linux/blob/master/kernel/sched/fair.c#L3063

Machine Learning for OS

Increased Automation

Approach	Goal	Implementations
Hardcode	Unchangeable	TCP_INIT_CWND
procfs/sysfs/netlink etc.	Update kernel values	ip route change ... initcwnd 10
eBPF	Update kernel code with user-written program	bpf_setsockopt(TCP_BPF_IW)
Machine Learning	Update kernel code with learned program from data	Kernel machine learning?

Machine Learning for OS

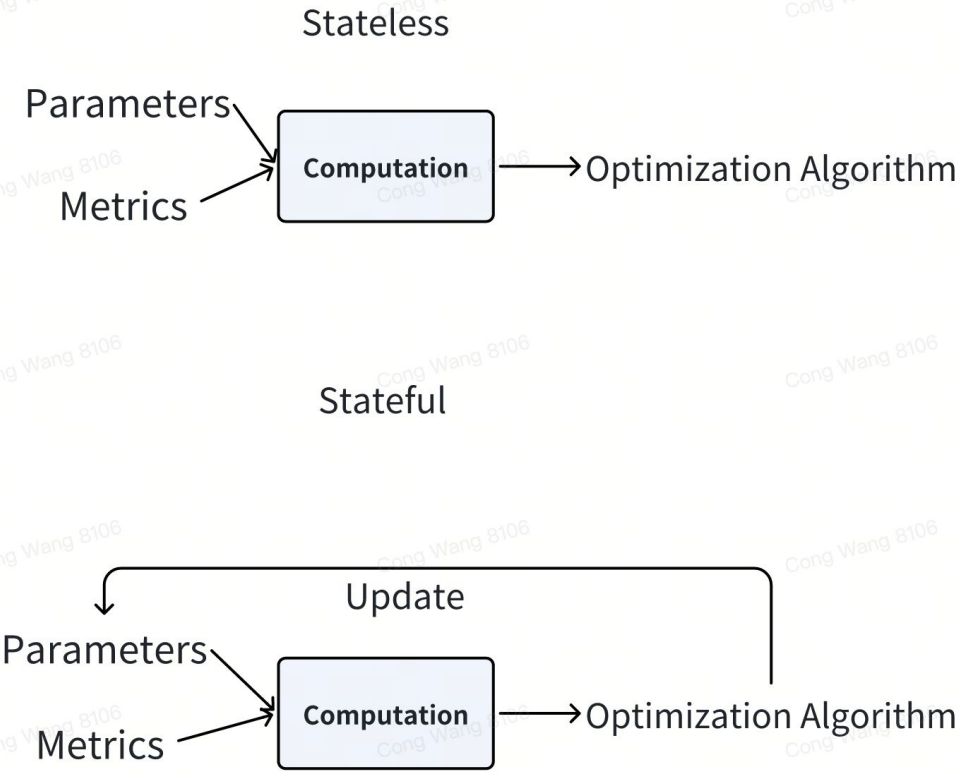
- Why: Machine Learning algorithms will help explore the potential of new solutions that rule-based approaches don't cover.
- How Machine Learning helps Operating System:
  - Prediction: based on the historical performance, predicts the future value.
  - Root cause analysis: look back into the historical data and apply retrospective analysis to diagnose potential cause.
  - Classification: classify problems into different categories.
  - Optimization: recommend best parameters to take for a specific workload.

Machine Learning Approaches for Optimization

- Problem definition
  - Definition: Find the best system parameters Xs to optimize the value of metric Y.
  - Parameters X: the system parameters that are tunable in the system.
    - Examples: TCP initial window, vm.watermark_scale_factor
  - Metric Y: the observable metric y that changes accordingly.
    - Examples: Service latency, Throughput, Resource utilization

- Machine learning approaches: Stateless & Stateful

Machine Learning Approaches for Optimization



Machine Learning Approaches for Optimization

- **Stateless** vs Stateful
  - Stateless: Stateless machine learning algorithms don't retain the memory of past inputs or iterations during the training process, and each prediction is made based solely on the current inputs.
  - Stateless optimization algorithms don't need to track previous evaluations of the objective functions to proceed with its search for optimum.
    - Example: Linear Regression/Random Search/Grid Search
    - When to use: simple scenario with very limited parameters tune; good for baseline setup, real time tuning.
    - Application: adaptivemm tuning (Linear Regression), nginx tuning - baseline setup (Random Search/Grid Search).

Machine Learning Approaches for Optimization

- Stateless vs **Stateful**
  - Stateful: Stateful machine learning algorithms maintain the form of state or memory across individual data points over time.
  - Specifically for optimization algorithms, they maintain and utilize the information from past evaluations of objective functions to guide the search for the optimal solution.
    - Examples: Bayesian Optimization/Genetic Algorithm/Simulated Annealing/Evolutionary Algorithm
    - When to use: non-real-time tuning, better result
    - Application: nginx tuning (Bayesian optimization)

Statistical Performance Evaluation

Data distribution comparisons

- To evaluate performance tuning results over the metrics, we utilize statistical tools (quantitative) in addition to data visualization (qualitative). We collect the metric results observed over the specified duration, and transform them into data distributions for statistical tools to compare.
- Ideally, the more dissimilar the two distributions are, the more influential the tuning is.
- When comparing data distributions, statistical approaches can be categorized into parametric tests and nonparametric tests. Parametric tests are based on assumptions about the distribution of population from which the sample was taken. Nonparametric tests are not based on such assumptions.

Statistical Performance Evaluation

Data distribution comparisons

- Parametric:
  - **Unpaired t-test**: a statistical test used to compare the means of two independent samples. It assumes that the samples are drawn from populations with approximately normal distributions and equal variances. The unpaired t-test examines whether the means of the two samples differ significantly.
  - **ANOVA**: a statistical test used to compare the means of three or more independent groups or treatments. It determines whether there are significant differences in means between the groups.

Statistical Performance Evaluation

Data distribution comparisons

- Nonparametric:
  - **Kruskal Wallis Test**: a nonparametric test used to determine if there are statistically significant differences between three or more independent groups.
  - **KS Test**: a non-parametric test that measures the "distance" between two samples; it compares the empirical cumulative distribution functions (CDFs) of the two samples.
  - **KL Divergence**: a measure of dissimilarity between two probability distributions. It quantifies how one distribution differs from a reference or target distribution.

Use Case - DAMON

- DAMON is a Linux kernel subsystem for memory access monitoring and optimization
- DAMON scheme has some parameters to specify target access pattern (size, age, and access frequency) of memory regions
- We used simple adapter to search for the best scheme for MySQL application. It runs different DAMON schemes and compares their performance using RSS and MySQL QPS metrics
- Applying DAMON in combination with zram to identify and page out cold memory achieves about 30% of memory reduction

Use Case - Nginx

- What to optimize: HTTP latency on NGINX server
- What to tune: 16 kernel sysctl parameters

Memory Management

- Swappiness
- Vfs_cache_pressure
- Dirty_ratio
- Min_slab_ratio
- min_unmapped_ratio

CPU Scheduler

- Sched_latency_ns
- Sched_min_granularity_ns
- sched_wakeup_granularity_ns

Networking

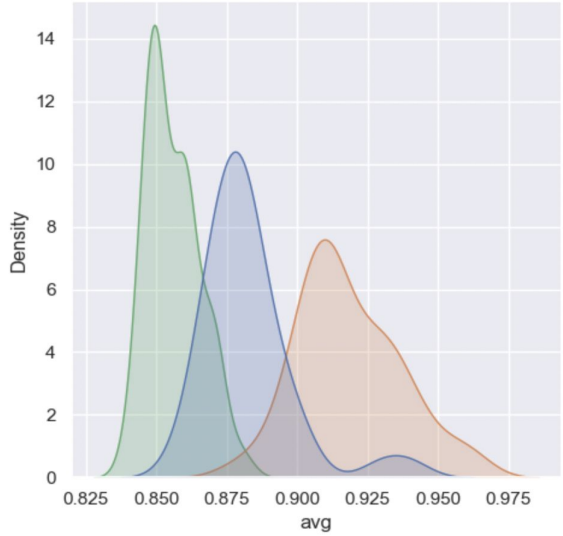
- Tcp_reordering
- Tcp_limit_output_bytes
- Tcp_notsent_lowat
- tcp_min_tso_segs

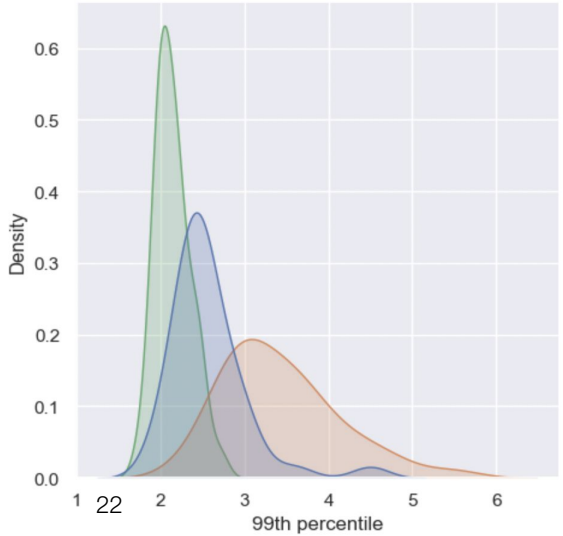
Block IO

- Fifo_batch
- Read_expire
- Write_expire
- writes_starved

Use Case - Nginx

- How to evaluate the performance:
  - each individual data point won't get a constant value due to random noise;
  - yet collective data points will form a data distribution across time, which is more convenient for observation and comparisons.





Data Distribution plots of avg and p99 of HTTP latency

X-axis: HTTP latency
Y-axis: density

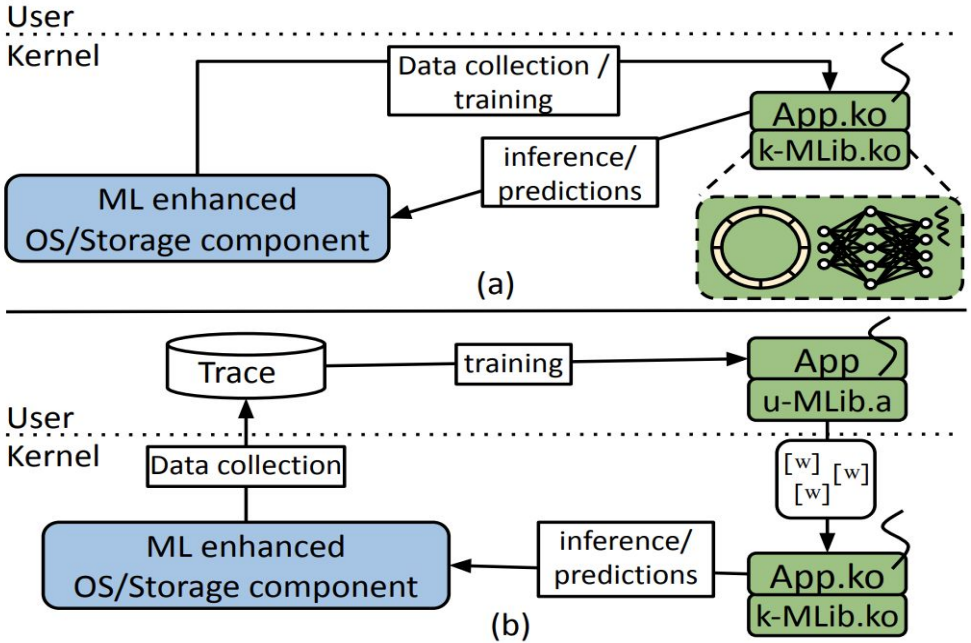
Legend:
Default Parameter Set
Parameter Set 1
Parameter Set 2

Use Case - Nginx

- The results of metric HTTP latency under different parameter sets
  - We utilize benchmarking tool wrk to launch tests for HTTP latency

No.	Description	Comments	Run Time per data point	HTTP Latency (ms)	Percentage Improvement
1	Expert Manual Testing (37)	NA	30 min	2.66	Baseline
2	Auto Tuning with Bayesian Optimization	Acquisition Function = UCB	17 min	2.97	-11.6 %
3		Acquisition Function = POI With prior knowledge	17 min	2.57	3.4 %
4		Acquisition Function = UCB With prior knowledge	17 min	2.46	7.5 %
5		Acquisition Function = EI With prior knowledge	17 min	2.44	8.2 %
6		Acquisition Function = EI With prior knowledge Better handling integer nature	17-20 min	2.34	12 %

Kernel Machine Learning



Source: “KML: Using Machine Learning to Improve Storage Systems”, https://arxiv.org/pdf/2111.11554.pdf

Kernel Machine Learning

- Kernel-space has more data to access, no data transfer to user-space.
- Kernel-space is more real-time hence more accurate.
- Kernel-space has floating point restrictions.
- Kernel-space has less tolerance for CPU or memory overhead.
- Linux kernel is written in C (or Rust).

Kernel Machine Learning

- Kernel disables FP to minimize context switching overhead.
- Quantization can help reduce computational and memory overheads, but it reduces accuracy.
- Fixed-point representation works within fixed ranges which can result in numerical instability.
- kernel_fpu_begin()/kernel_fpu_end() are already used in RAID and crypto code.

Kernel Machine Learning

- Page prefetching
- I/O scheduling
- CPU load balance
- Caching eviction
- Network congestion control
- Intrusion detection

Conclusions

- We envision machine learning is important for OS optimizations.
- Although there are limitations, we believe that kernel machine learning is not only possible but also necessary.

Acknowledgement

- Jasmine Mou
- Krz Sywula
- Bobby Eshleman
- Yaxin Chen

THANKS

.