Linux Kernel Autotuning

Cong Wang, Jasmine Mou
System Technologies & Engineering
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?

Traditional Programming

Data → Computation → Output

Program

Machine Learning

Data → Computation → Program

Output
Why Machine Learning?

Features X → Learner → Model

X_{new} → y_{predicted}
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
/ * It's the expected callback index, assume sequential access. 
  * Ramp up sizes, and push forward the readahead window. 
  */ 
expected = round_up(ra->start + ra->size - ra->async_size, 
                   1UL << order); 
if (index == expected || index == (ra->start + ra->size)) { 
  ra->start += ra->size;  
  ra->size = get_next_ra_size(ra, max_pages);  
  ra->async_size = ra->size;  
  goto readit; 
}
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
/*
 * If a workload spans multiple NUMA nodes, a shared fault that
 * occurs wholly within the set of nodes that the workload is
 * actively using should be counted as local. This allows the
 * scan rate to slow down when a workload has settled down.
 */

g = deref_curr_numa_group(p);
if (!priv && !local && g && g->active_nodes > 1 &&
numa_is_active_node(cpu_node, ng) &&
numa_is_active_node(mem_node, ng))
    local = 1;

/*
 * Retry to migrate task to preferred node periodically, in case it
 * previously failed, or the scheduler moved us.
 */
if (time_after(jiffies, p->numa_migrate_retry)) {
    numa_placement(p);
    numa_migrate_preferred(p);
}
## Machine Learning for OS

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

Increased Automation
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 $X_s$ 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

Stateless

Parameters

Metrics

Computation → Optimization Algorithm

Stateful

Update

Parameters

Metrics

Computation

Optimization Algorithm
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)
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_minGranularity_ns
- sched_wakeupGranularity_ns

### Networking
- Tcp_reordering
- Tcp_limit_output_bytes
- Tcp_notsent_lowat
- tcp_min_tso_segs

### Block I/O
- 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.
Use Case - Nginx

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

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

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