CAPES: Unsupervised System Performance Tuning Using Neural Network-Based Deep Reinforcement Learning

Yan Li, Kenneth Chang, Oceane Bel, Ethan L. Miller, Darrell D. E. Long
{yanli,kchang44,obel,elm,darrell}@ucsc.edu
Storage Systems Research Center
University of California, Santa Cruz

This research was supported in part by the National Science Foundation under awards IIP-1266400, CCF-1219163, CNS-1018928, CNS-1528179, the Department of Energy under award DE-FC02-10ER26017/DESC0005417, by a Symantec Graduate Fellowship, by a grant from Intel Corporation, and industrial members of the Center for Research in Storage Systems.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the researchers and do not necessarily reflect the views of the National Science Foundation, the Department of Energy, or other sponsors of this project.

The problems

- Large-scale high-performance computer storage systems play an important role in modern computing technologies:
  - Data centers
  - High-Performance Computers
  - Storage for enterprise computing
- Their performance can degrade dramatically during peak hours.
- Traditional tuning methods are costly because they require:
  - Human experts
  - Lengthy tune-benchmark-tune cycles
  - Fine tuning for each workload

Deep Q-Learning (DQL)

- Reinforcement learning concerns with how an agent ought to take actions in an environment in order to maximize a cumulative reward.
- DQL is reinforcement learning that uses Q-function to calculate reward and deep neural networks as value function.
- Q-function: the maximum discounted future reward when performs perfectly.

$$Q(s_t, a_t) = \sum_{i=t}^{n} \gamma^{i-t} r_t$$

(s_t is system state at time t, a_t is action at time t, r_t is reward at time t, γ is reward discount.)

- Q can be solved iteratively (Bellman's equation), which indicates that Q-learning can converge:

$$Q(s,a) = r + \gamma \max_{a'} Q(s', a')$$

Advantages

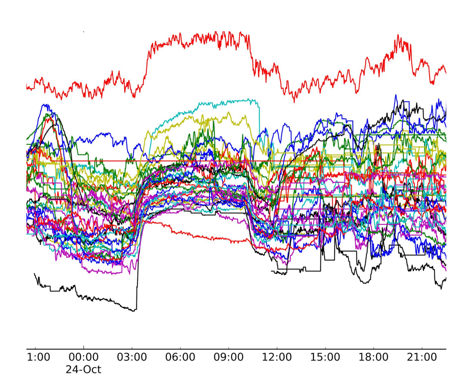
End-to-end coverage:
- Controls congestion at client, network, servers, and drives.

Improve throughput or fairness during congestion:
or both at the same time!

Fully automatic and requires little human effort:
- Online training and real-time online tuning, good at tuning dynamic and changing workloads.
- Faster than human tuning cycles.
- Works with any parameters of any systems in theory.

Our solution: Deep Q-Learning for traffic control

- Uses *congestion window size* and *rate limit* to control storage traffic.
- Uses fully automatic, unsupervised, Deep Neural Network-based reinforcement learning to turn parameter values.



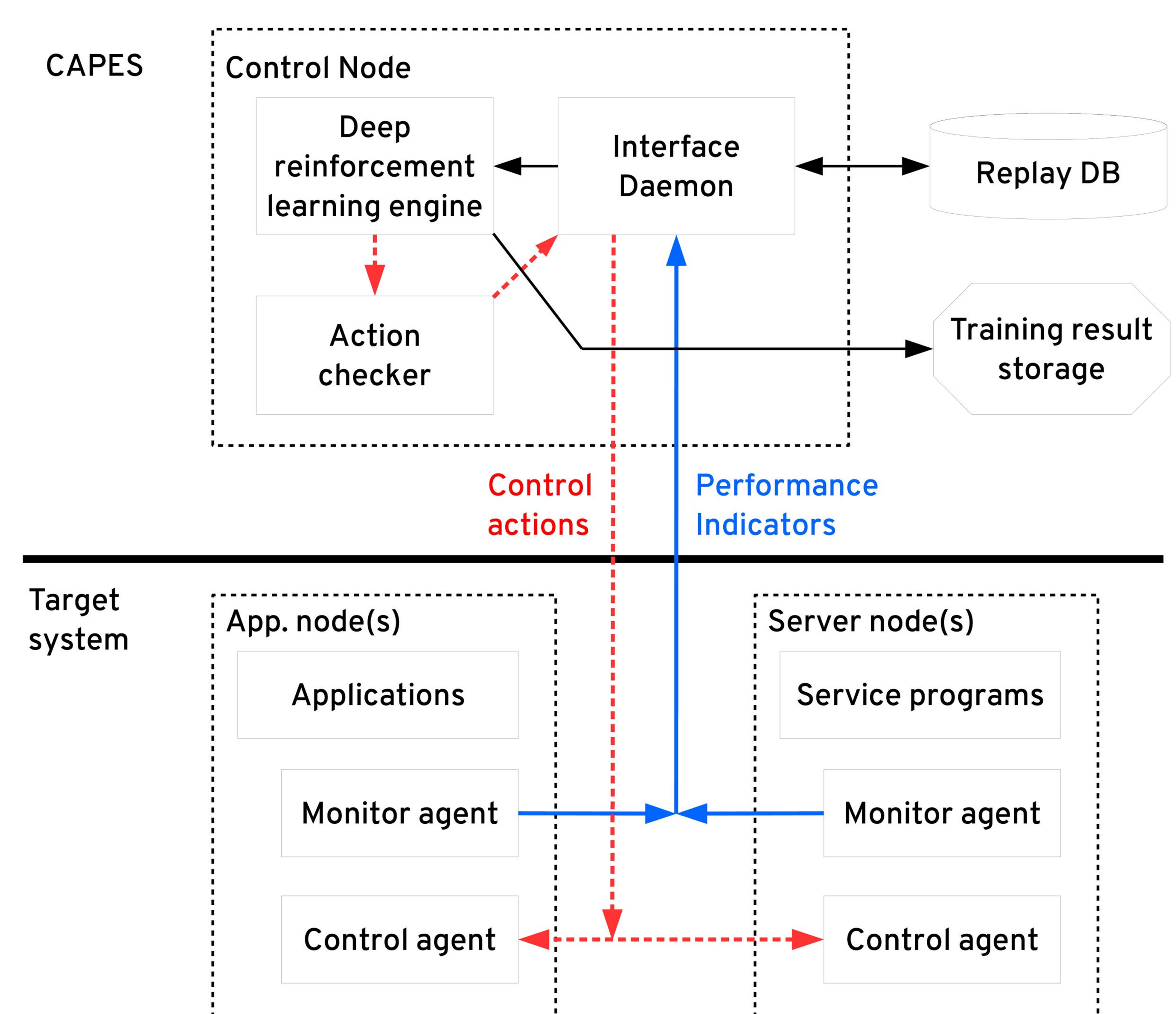
Unfiltered raw performance indicators (system state) → Value function →

Candidate action	Predicted reward
Action 0: Do nothing	0.382
Action 1: Increase Parameter A	0.741
Action 2: Decrease Parameter A	0.127
Action 3: Increase Parameter B	0.547
Action 4: Decrease Parameter B	0.123
Action 5: Increase Parameter C	0.372
Action 6: Decrease Parameter C	0.457

Action table (possible actions)

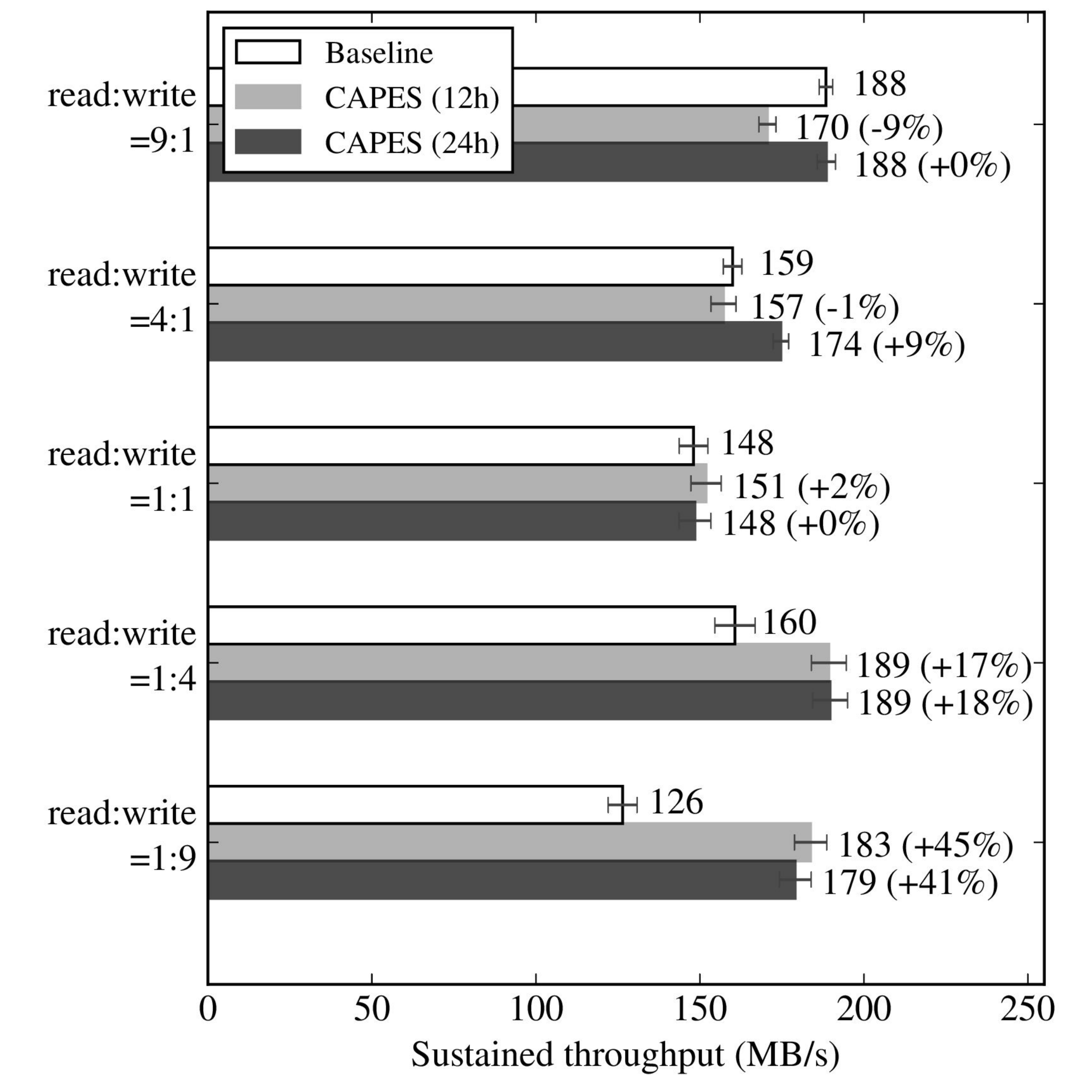
- Deep Q-Learning is an online reinforcement learning method, in which the system automatically explores tuning actions, collect rewards, and uses DNN to extrapolate to unseen situations.
- Input to the system is *Performance Indicators*, which include all measurement data of the target system.
- A set of predesignated *actions* decide how to tweak parameters.
- In each tick, the Value function (DNN) calculates which action has the highest anticipated *reward* and performs that action.

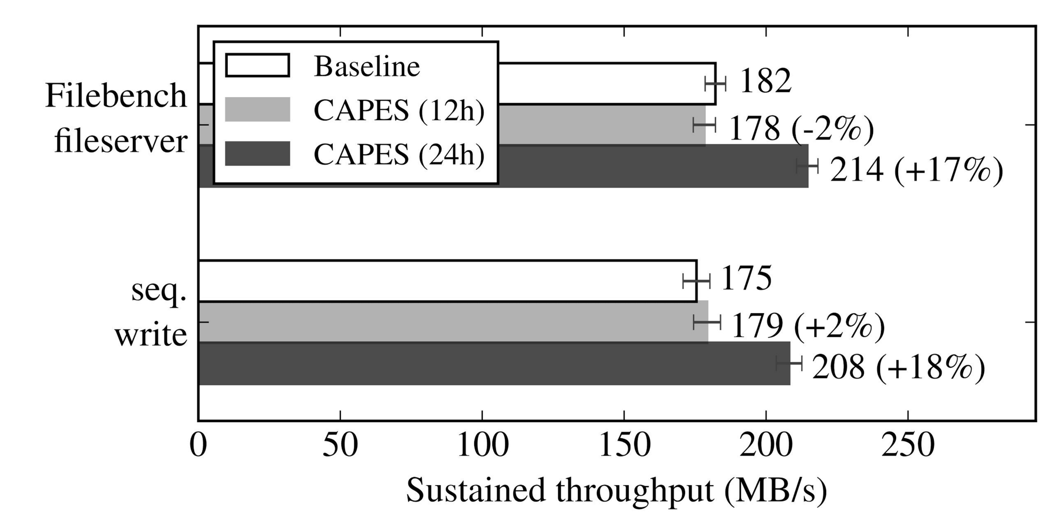
CAPES Design



Performance indicators	Definition
write throughput	the write throughput of the past second
read throughput	the read throughput of the past second
ack_ewma	exponentially weighted moving average (EWMA) of gaps between RPC acks
send_ewma	EWMA of gaps between sender timestamp embedded in RPC acks
pt	the time needed for server to finish reading/writing 1 MB data request
pt_ratio	current pt / min(pt) seen so far
dirty bytes in write cache	the dirty bytes on the client write cache

Prototype and Evaluation on Lustre





- Implemented using TensorFlow running on GPU.
- DNN has 2 hidden layers and uses Adam optimizer.
- Exploration period is 2 h.
- 4 servers + 5 cilents with multiple threads to saturate the bandwidth.
- CAPES control server uses GPU
- Graph shows throughputs at saturation point.
- Tuning congestion window size has a larger effect on write-heavy workloads.
- 12 h training time is enough for simple workloads (1st fig), and 24 h is needed for complex workloads (2nd fig).

Conclusion and Future Work

- Minimally intrusive, can be deployed to production system.
- Fully unsupervised, model-less tuning.
- Online training and responsive to changing workloads.
- Effect on a wide range of workloads, especially write heavy workloads.
- Suitable for any deployment before hitting the scale-out bottleneck.
- For details of CAPES, such as full algorithm, overfitting test, and prediction error over time, refer to paper "CAPES: Unsupervised Storage Performance Tuning Using Neural Network-Based Deep Reinforcement Learning." SC'17.
- Source code will be available at: https://github.com/mlogic/capes-oss

Version: 10/3/2017