Reinforcement learning concerns with how an agent Q-learning, online training and real-time tuning, good at controls congestion at client, network, servers, and faster than human tuning cycles. Works with any parameters of any systems in theory.

Online training and responsive to changing workloads, suitable for any deployment before hitting production system. Fully unsupervised, model-less tuning. 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.

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.
- 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/mllogic/capes-oss

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 performed perfectly.
  \[ Q(s, a) = \sum_{t} \gamma^t r_t \]
  \( s \) is system state at time \( t \), \( a \) is action at time \( t \), \( r_t \) is reward at time \( t \), \( \gamma \) 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.