ECML PKDD
Dublin, Ireland
10-14 SEPT 2018

On Optimizing Operational Efficiency in Storage Systems via Deep Reinforcement Learning

Sunil Srinivasa[1], Girish Kathalagiri[1], Julu Subramanyam Varanasi[2], Luis Carlos Quintela[1], Mohamad Charafeddine[1], ChiHoon Lee[1]

[1] Samsung SDS America

[2] Samsung Semiconductors Inc

I. Motivation, Intro, & Problem Setup

II. Short Primer on RL

III. Problem Formulation

IV. Actor Critic Network & Algorithm

V. Results

Motivation

- Data Centers as large sinkholes of energy.
- US data centers consumption is about 70 billion Kwh, 1.8% total US energy consumption (Lawrence Berkeley Lab study 2016), ~ $8B

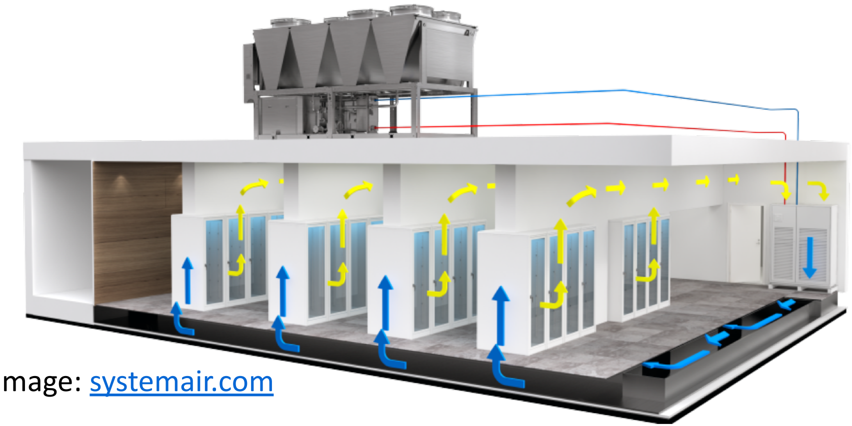


Image: systemair.com

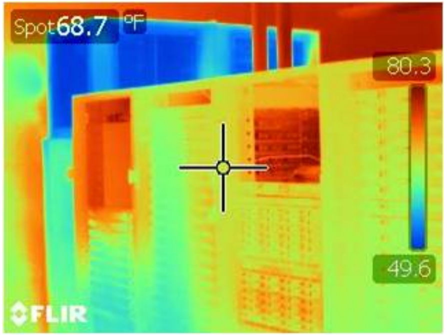
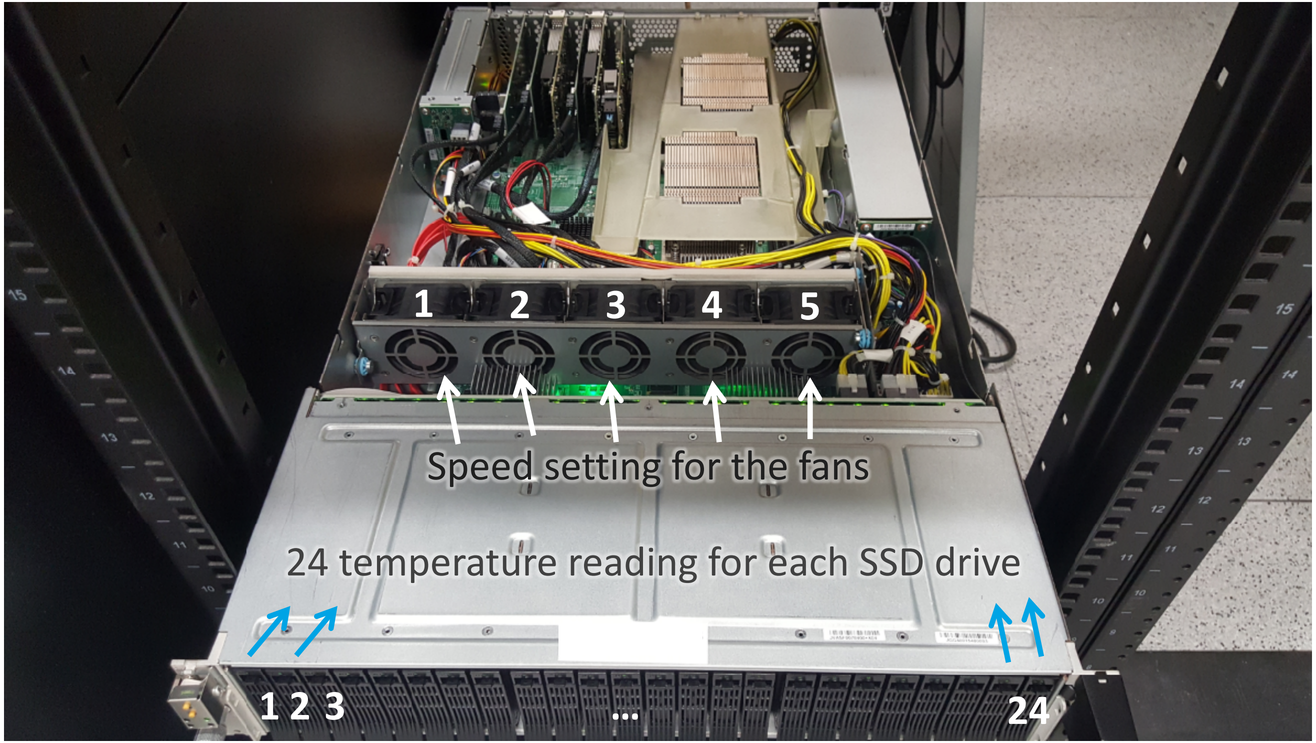


Image: *www.aashe.org*

Building-level: Data Center energy usage optimization, optimizing water pumps, cooling towers, ..

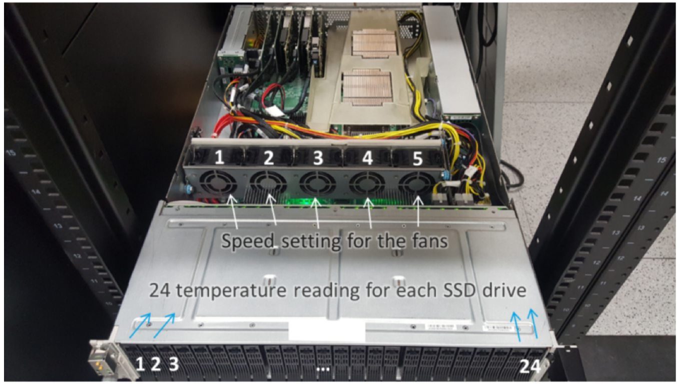
Server-level: energy usage optimization per server, optimizing fans speed inside servers
focus of this talk

Problem setup: operational efficiency of a storage server



24 SSDs solid state drives for storage

Problem setup: operational efficiency of a storage server



- High temperatures lower the performance of reading/writing
- High temperatures shorten the lifespan of the SSD

- Temperature varies based on the workload: read or write, how many KB, and how often.

- Current status quo control is rule-based.

We would like to use Deep Reinforcement Learning to represent the state of operation and learn the optimum policy for controlling the fan speeds, as a proxy of energy usage for cooling.

Contributions

- **Model-free approach**, no need to understand to SSD server behavior dynamics of workload and temperature
- We **trained on the real environment** (the server), not through a simulator
- We **designed a Reward function for this problem** to guide the algorithm towards the desired operation behavior (servers temperature & fans speeds)

I. Motivation, Intro, & Problem Setup

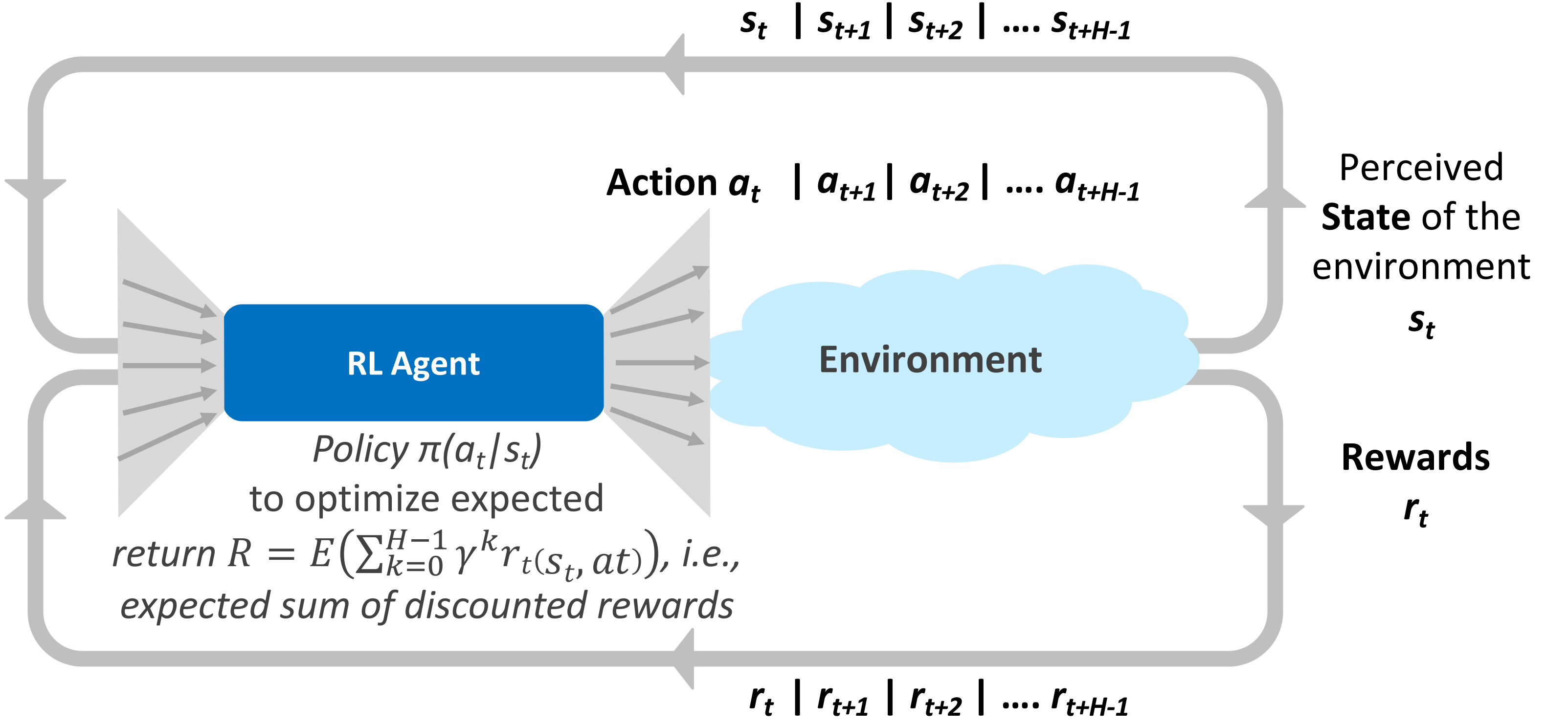
II. Short Primer on RL

III. Problem Formulation

IV. Actor Critic Network & Algorithm

V. Results

Reinforcement Learning



Reinforcement Learning: Action Value, State Value, Advantage functions

- **Action Value Function $Q^{\pi}(s_t, a_t)$**: is the scalar **expected long-term return** achieved for following **Policy π** when in **state s_t** by performing **action a_t**. → How good is this state-action pair.

- **State Value Function $V^{\pi}(s_t)$:** is the scalar **expected long-term return** achieved for following **Policy π** when in **state s_t** averaged over all possible actions $\{a\}$ from that state. → How good is this state s_t.

- **Advantage Function $A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$** is the advantage of taking action a_t when in state s_t compared to the average of all possible actions from that state.

The Action and Value functions are unknown, and we need to **learn them by interacting with the environment**. One approach is through training Deep Neural Network(s) which takes the state as an input and outputs an approximation of Q and V.

I. Motivation, Intro, & Problem Setup

II. Short Primer on RL

III. **Problem Formulation: State, Action, Rewards**

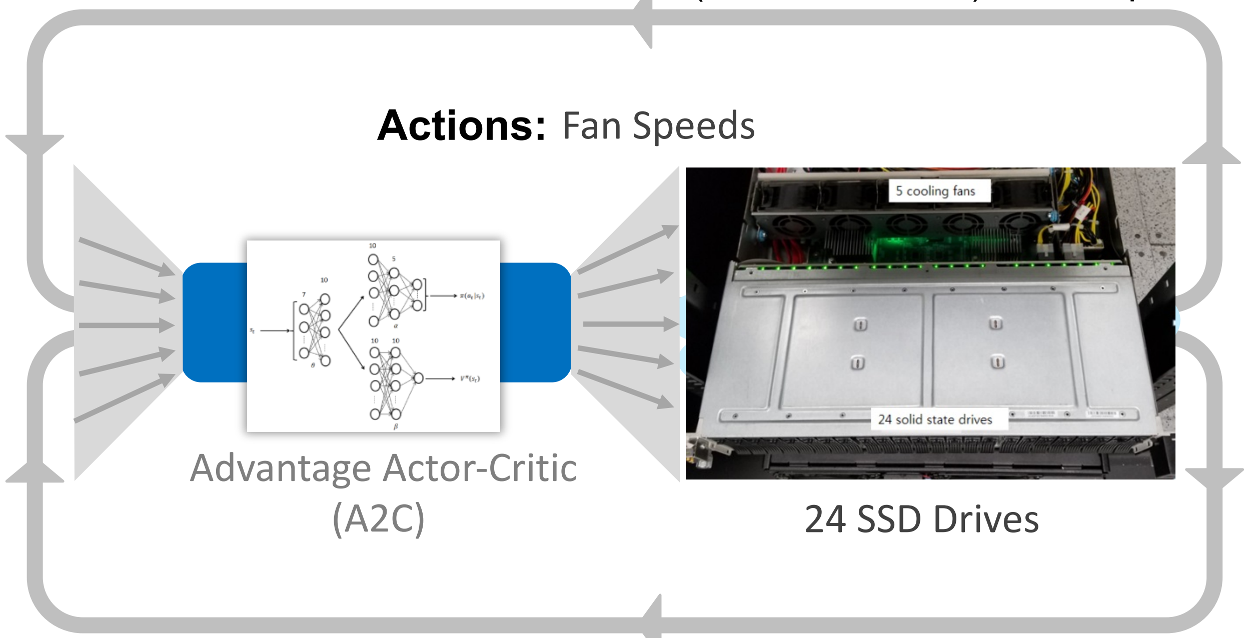
IV. Actor Critic Network & Algorithm

V. Results

Problem setup using Deep Reinforcement Learning

State: function of workload (Reads or Writes) and temperature

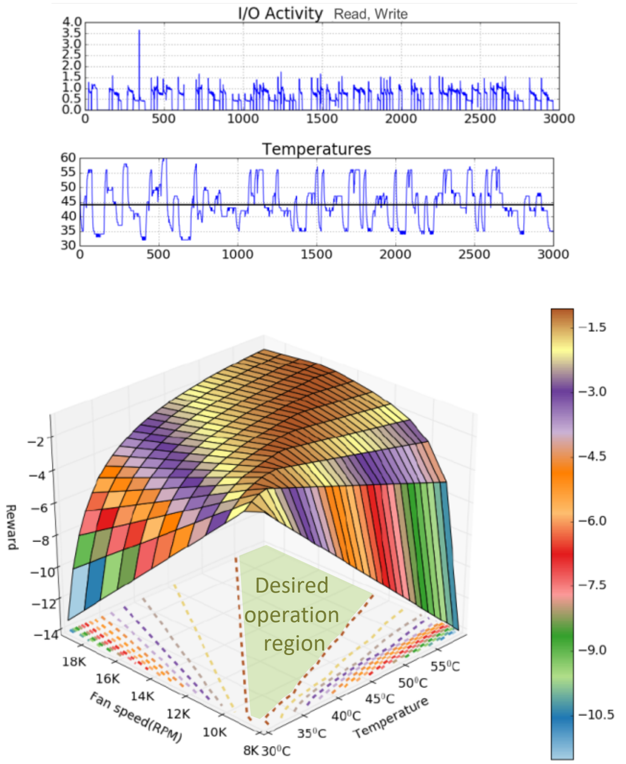
Actions: Fan Speeds



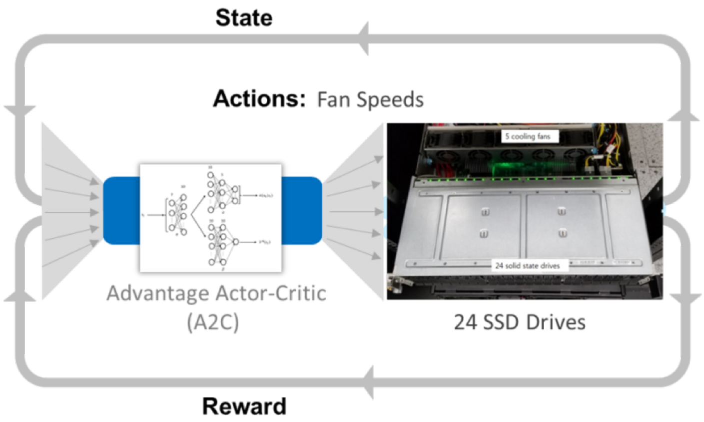
Advantage Actor-Critic (A2C)

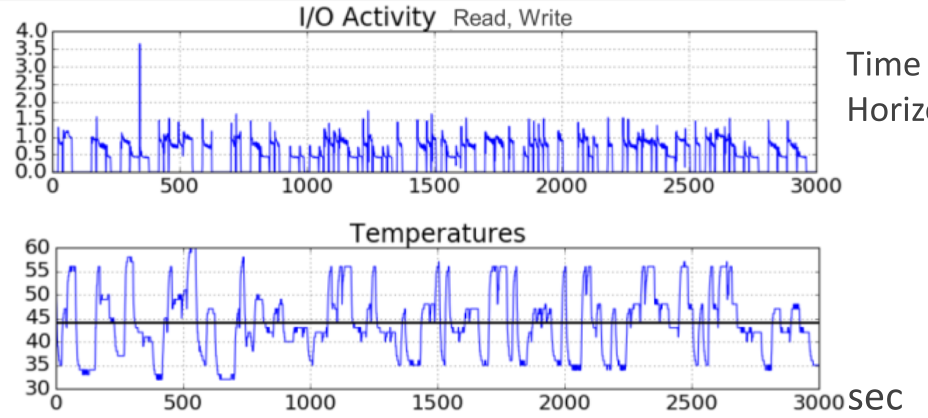
24 SSD Drives

Reward: A function of Temperatures & Fans Speeds



Problem Formulation: State





Time slot = 25 sec
Horizon = 10 slots

State =

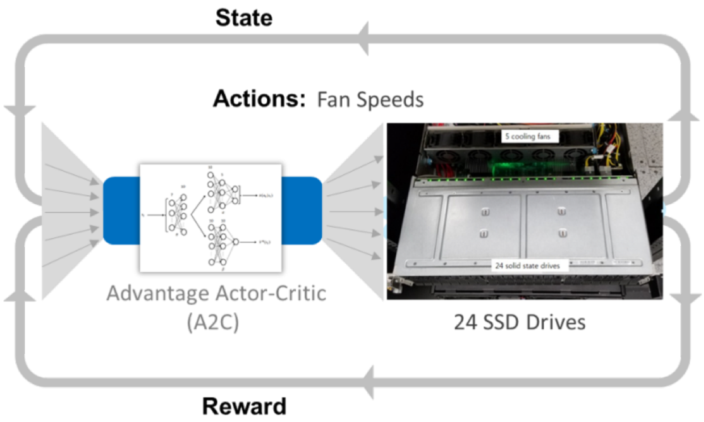
1. $\Gamma($**# of I/O requests** to the server **per second**$)$
2. $\Gamma($**# of Kbytes read**, averaged over all SSDs per sec$)$
3. $\Gamma($**# of Kbytes written**, averaged over all SSDs per sec$)$
4. $\Gamma($**# of Kbytes read** in **previous time slot**, averaged over all SSDs$)$
5. $\Gamma($**# of Kbytes written** in **previous time slot**, avg. over all SSDs$)$
6. $\Gamma($**Mean Temperate**: averaged over all SSDs$)$
7. $\Gamma($**Mean Fan Speed**: averaged over all 5 cooling fans$)$

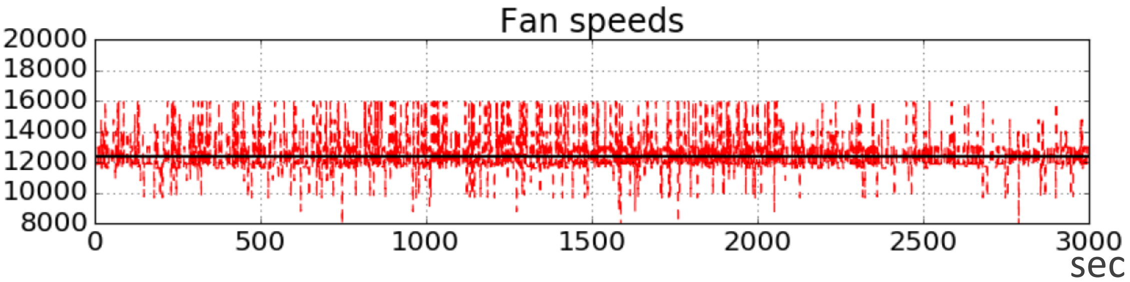
Read via `iostat` (items 1–5)

Read via `ipmi-sensors` (items 6–7)

Where normalization operation per field is $\Gamma(x) = \dfrac{x - minX}{maxX - minX}$

Problem Formulation: Actions





We tried 2 approaches

Raw Actions

Size of Actions space = **7**

A = {6, 8, 10, .., 18}

Kilo rpm (revolutions per minute)

Incremental Actions

Size of Actions space = **3**

A = {Decrease by 1000,

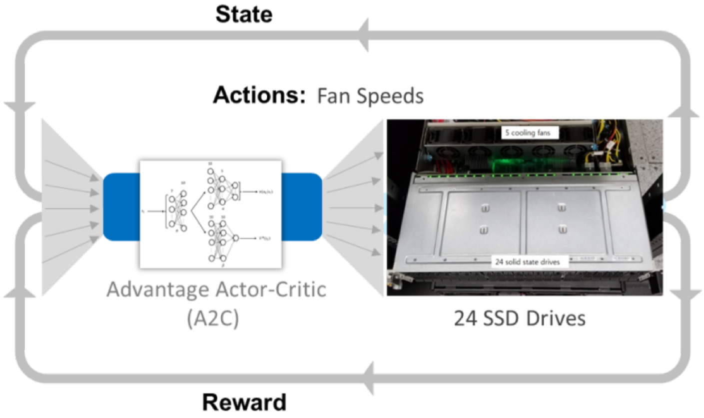
No change,

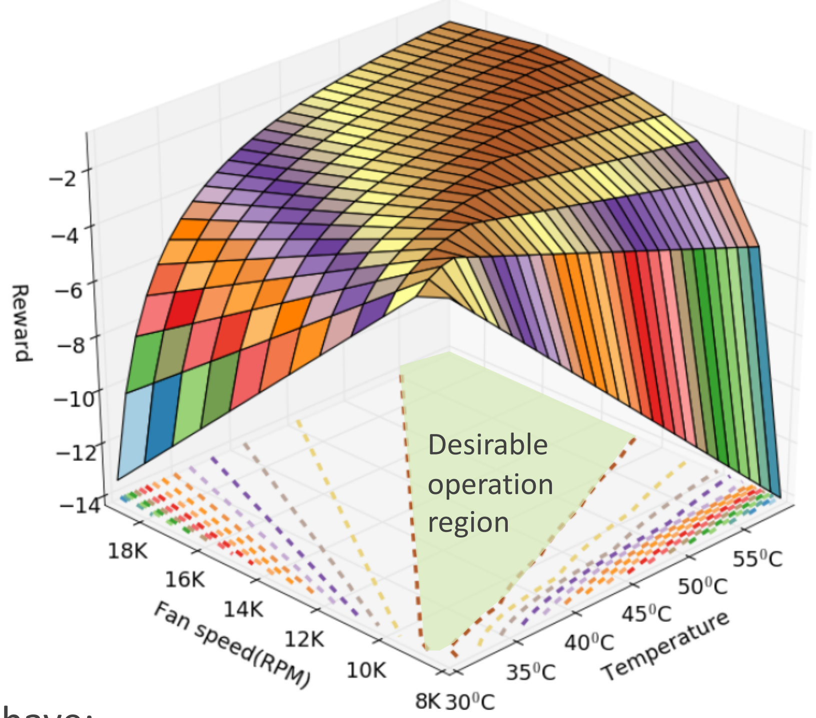
Increase by 1000} rpm

Allows for smoother transitions

Action is applied to all 5 fans via the `ipmitool` command

Problem Formulation: Rewards





Reward shaping is an important design step which requires domain knowledge and guides the RL agent towards its optimal behavior

For this RL application, a good reward function should have:

1) High reward for low fan speeds and low temperature
2) Low reward for high fan speeds and low temperature – as this is wasting energy
3) Low reward for low fan speed and high temperature – as this will damage the SSDs

$$R = -max\left(\frac{\Gamma(T)}{\Gamma(F)}, \frac{\Gamma(F)}{\Gamma(T)}\right)$$

We chose this design above.
T is the mean temp across all SSDs.
F is the mean speed across fans.

SAMSUNG SDS SAMSUNG

I. Motivation, Intro, & Problem Setup

II. Short Primer on RL

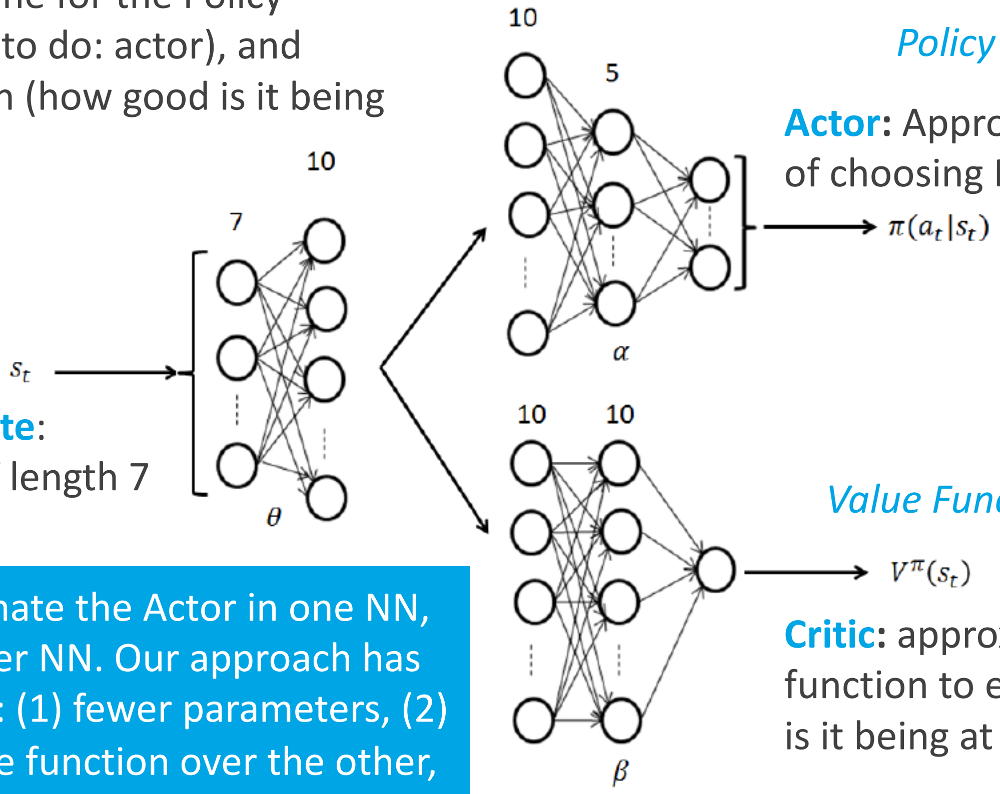
III. Problem Formulation

IV. Actor Critic Network & Algorithm

V. Results

Actor Critic (A2C) Dueling Network Architecture

A NN with 2 branches: one for the Policy evaluation (what action to do: actor), and one for policy estimation (how good is it being in such state: critic)



State:
Vector of length 7

Policy Neural Network

Actor: Approximates optimal policy, of choosing Best action for state ***s***

Value Function Neural Network

Critic: approximates the state-value function to estimate how attractive is it being at state ***s***

Other option is to estimate the Actor in one NN, and the Critic in another NN. Our approach has the advantage of having: (1) fewer parameters, (2) mitigates overfitting one function over the other, (3) faster convergence, and (4) more accurate approximations for the actor & the critic.

A2C algorithm pseudo-code

Algorithm 1 Advantage Actor-Critic (A2C) - pseudocode

// Notate shared parameters by θ and actor- and critic-specific parameters by α and β, respectively.
// Assume same learning rates η for θ, α as well as β. In general, they may all be different.
Initialize θ, α and β via uniformly distributed random variables.
repeat
  Reset gradients $\mathrm{d}\theta = 0$, $\mathrm{d}\alpha = 0$ and $\mathrm{d}\beta = 0$.
  Sample N trajectories $\tau_1, \ldots, \tau_N$ under the (current) policy $\pi(\cdot; (\theta, \alpha))$.
  i = 1
  repeat
    $t_{\text{start}} = t$
    Obtain state s_t
    repeat
      Perform action a_t sampled from policy $\pi(a_t | s_t; (\theta, \alpha))$.
      Receive reward $r_t(s_t, a_t)$ and new state s_{t+1}
      $t \leftarrow t + 1$
    until $t - t_{\text{start}} = H$
    $i \leftarrow i + 1$
    Initialize R: $R = V(s_t; (\theta, \beta))$
    for $i \in \{t-1, \ldots, t_{\text{start}}\}$ **do**
      $R \leftarrow r_i(s_i, a_i) + \gamma R$
      Sum gradients w.r.t θ and α: // gradient ascent on the actor parameters
      $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \nabla_\theta \log \pi(a_i | s_i; (\theta, \alpha)) \, (R - V(s_i; (\theta, \beta)))$
      $\mathrm{d}\alpha \leftarrow \mathrm{d}\alpha + \nabla_\alpha \log \pi(a_i | s_i; (\theta, \alpha)) \, (R - V(s_i; (\theta, \beta)))$
      Subtract gradients w.r.t β and θ: //gradient descent on the critic parameters
      $\mathrm{d}\theta \leftarrow \mathrm{d}\theta - \nabla_\theta \, (R - V(s_i; (\theta, \beta)))^2$
      $\mathrm{d}\beta \leftarrow \mathrm{d}\beta - \nabla_\beta \, (R - V(s_i; (\theta, \beta)))^2$
    end for
  until $i = N$
  // Optimize parameters
  Update θ, α and β: $\theta \leftarrow \theta + \eta \mathrm{d}\theta$, $\alpha \leftarrow \alpha + \eta \mathrm{d}\alpha$, $\beta \leftarrow \beta + \eta \mathrm{d}\beta$.
until convergence.

I. Motivation, Intro, & Problem Setup

II. Short Primer on RL

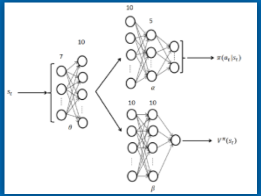
III. Problem Formulation

IV. Actor Critic Network & Algorithm

V. Results

Learning... over few days

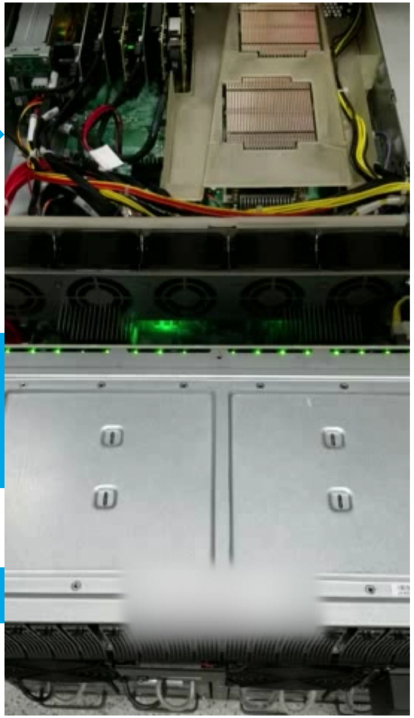
RL Agent



Fan Speeds

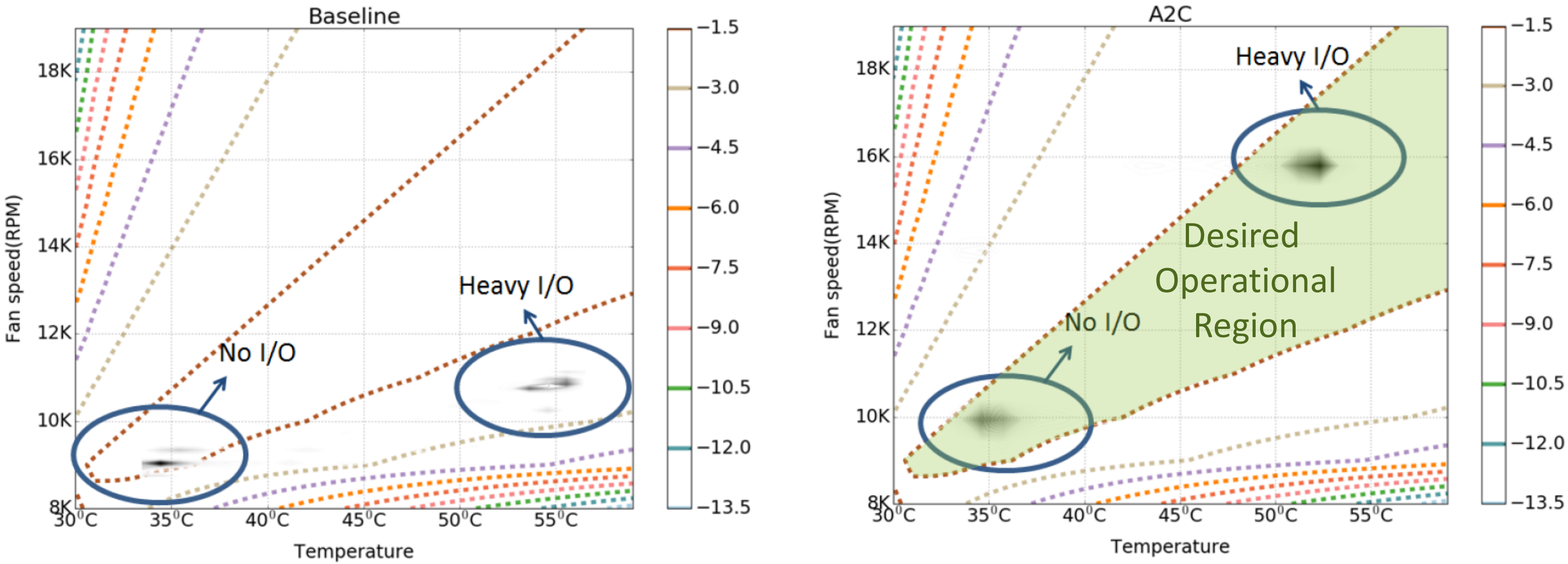
State: Different workloads, Temperature

Reward



- Learning directly on the real environment (no simulator)
- Model-free: does not require any knowledge of the SSD server behavior dynamics
- Exposed to different stochastic workloads

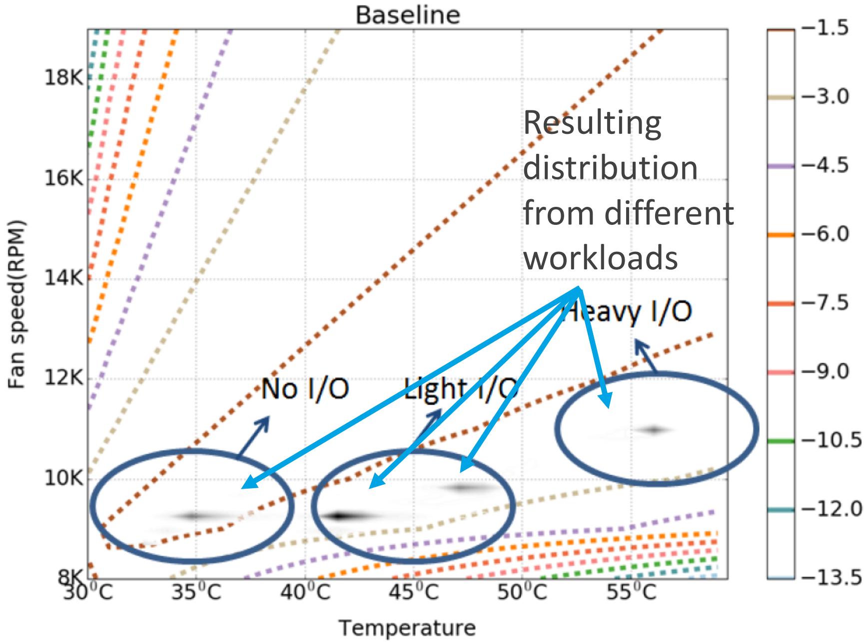
Performance for Idle Vs Heavy Periodic I/O workloads on the operational contours



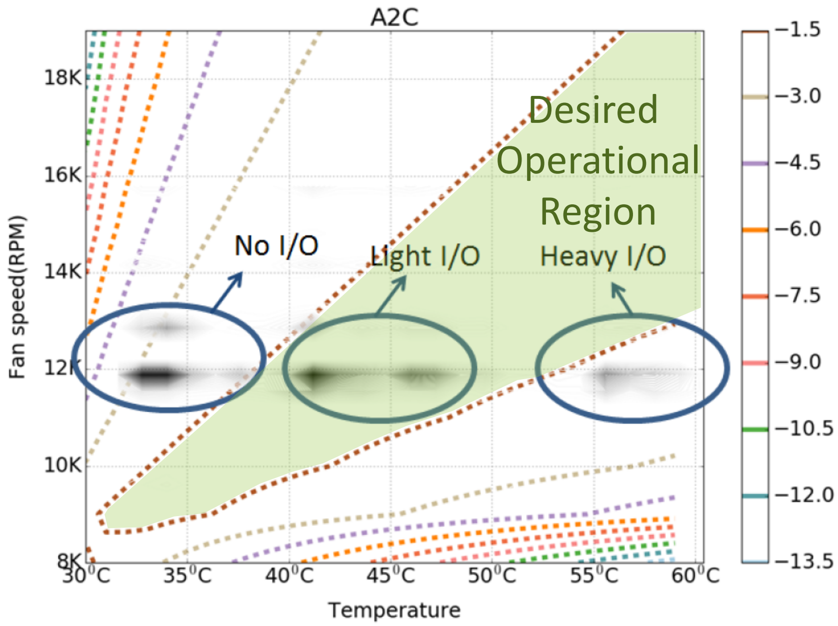
Status Quo controller

Using Deep Reinforcement Learning with Raw Actions

Performance for different stochastic workloads – Attempt 1

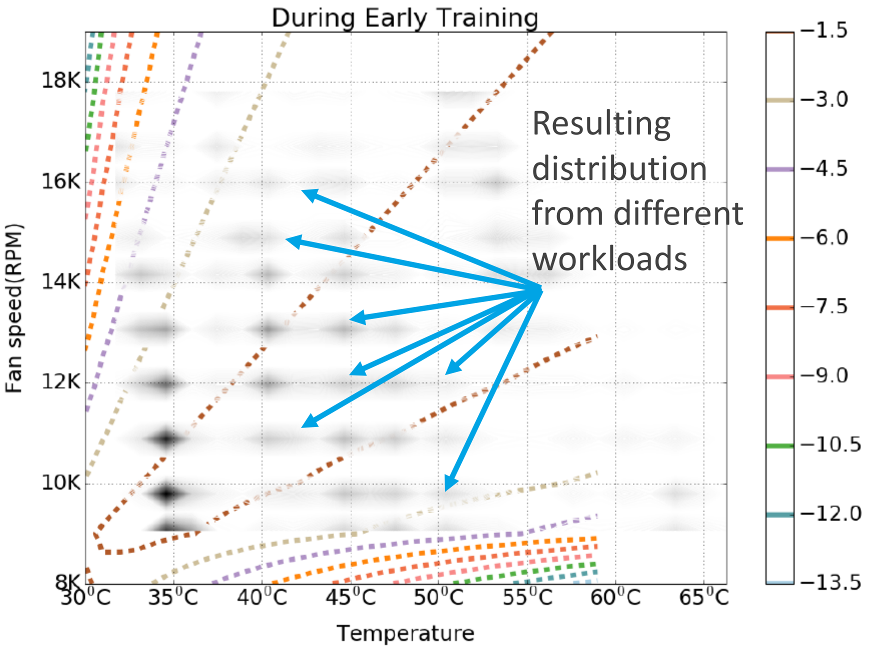


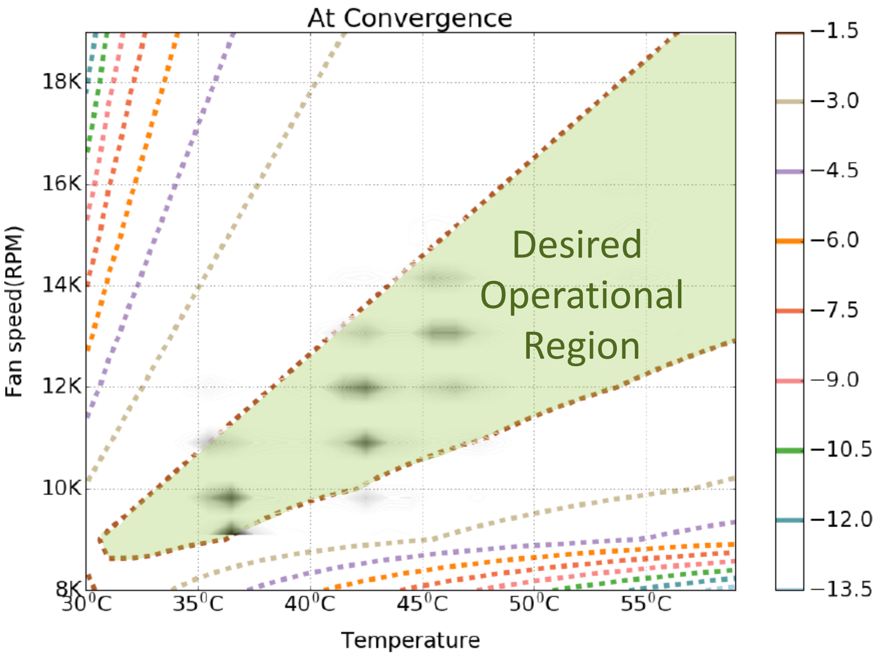
At the beginning of training, algorithm is exploring and learning



Using Deep Reinforcement Learning with Raw Actions – Suboptimal performance, likely due to insufficient exploration

Performance for different stochastic workloads – Attempt 2





At the beginning of training, algorithm is exploring and learning

Once finished learning right policy, operational behavior is within desired region. Used DRL with incremental actions.

insight to !nspiration

SAMSUNG SAMSUNG SDS