More

Control

Loops (maybe)

Final Project

- Expectations

- Some things to keep in mind

- Progress?

Briefly return to last week's art
Automatic Reliability Testing

"How To Evaluate The Correctness Of Control Loops"

Several problems

What does it mean for a control loop to be correct?

- Satisfies utility function?
  - How to automatically check?
  - Over what time period?
  - ...

Aside: How do people do this even manually?

- Model the system in one of several ways
  - Fluid model: use tools from fluid dynamics
  - Queuing theory
  - Net. calculus
  - ...

But requires manual effort
  - To select what to model
  - To actually model things
(1) Can correctly change the system
   - Correctly increase or decrease resource
   - Correctly affect request scheduling
   - ... 

Hmm, but how to figure out what the control loop wants to do?

How would you do it (e.g., when debugging an algorithm)?

ART way: Based on observation about how
K8S works (also how Borg works —
This ties into what comes next)
Core Idea
- Produce different seq. of inputs to drive control loops to different decisions
  \( \rightarrow \) Report bug if control loop crashes due to seq. of inputs.
- Check decisions are correctly implemented
  \( \rightarrow \) Report bug if not

But, is this enough to be ‘confident’ about control loops?
Auto scaling

1. Resources required
   - App to run
   - Alice
   - App Developer

2. ORCHESTRATOR
   - Allocate resources

... ...

Carolyn
- Cluster Admin

Observes utilization, decides when to get more stuff

Tension

Voice:
- cheap!

Wants application to run fast, most likely with lots of resources

Wants to reduce cost, most likely with fewer resources

One way to solve

But how much to change? How much to allocate?
AWS 2011: Average CPU utilization ≤ 10%
Azure 2016: Average CPU utilization 15-40%
Alibaba 2018: CPU median < 40%, p90 < 50%
Memory median — FG: 30%, p90: 40%, BG: 90%, 85%

Quickly leads to the idea of autoscaling
Allocate & deallocate resources on demand
Can be more efficient

Like many obvious ideas, this one is very hard in practice

Vertical Scaling

Horizontal Scaling

- More CPU/
  memory/
  disk space/
  o...
Some challenges

- Effects take a while to become visible
  
  Horizontal: Initialization
  
  Vertical: Kind of like initialization

  Don't wait too long to scale!!

- Effects are non-linear, might not even be a smooth function

Steady state capacity

Be careful about how much to scale by each time

- Performance & utilization metrics might be unreliable around auto-scaling events
- Be careful when to measure!
- Don't scale too often! [Hysteresis]
- Horizontal scaling may not be safe
  - Databases
  - Vertical scaling may not be safe

Autoscaling is too hard! Let us have someone else do it.

- Control loop
Q. Inputs

→ Must be careful about when to measure
→ Must be careful about what to measure

Low tenk/High resp.
Low Latency

→ Must be careful about how to measure.

→ Averaging is the big hammer, everyone uses here to get rid of short term effects!

Q. How often should the control loop change things

- Hysteresis

Q. How much should the control loop change allocations by?
Building a control loop is hard. Let us have a machine learn it?

- How to convince users it is safe?

  - Google: ensemble of simple models

  => simple: weigh one of the measured quantities more heavily

  Argument: Easy for a developer to see what feature was found important

But really: Even normal control loops are hard to understand. Best we can do is compare against expectation
What Google does

What one can?

SUM it RL vs Control Loops (FIRM)

- First: Not a real distinction

  Ben Recht & Others: RL is just fancy control theory

  \[ \text{Use feedback (critic) to decide whether action was appropriate} \]
So what is different

- More freedom on how to interpret/weight signals
- More freedom on choosing actions
- Weights of actions can change due to feedback
  "sort of a control loop for a control loop."

What is with the SVM bit?

- Two steps
(1) Use SVM to pick target footprint
(2) Use RL to auto-scale target

Why split?

→ Space of possible choices to consider

D52 ° Does application knowledge help

Unsurprising: Yes

Surprising: Tricks to model & capture information

Requires several assumptions beyond application knowledge.