'MORE'

CONTROL

LOOPS (maybe)

Final Project

○ Expectations

○ Some things to keep in mind

○ Progress?

BRIEFLY RETURN TO LAST WEEK: 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?

→ ...

Bottom line: Hard? Needs more work?

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

→ ...

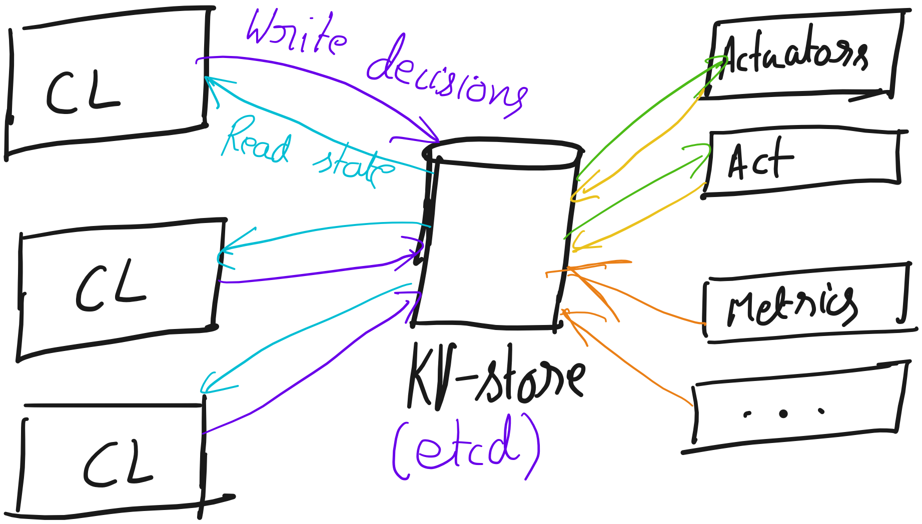
(b) Can correctly change the system

- ↳ Correctly increase or decrease resources
- ↳ Correcty 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

  ↳ Report bug if control loop crashes due to seq. of inputs.

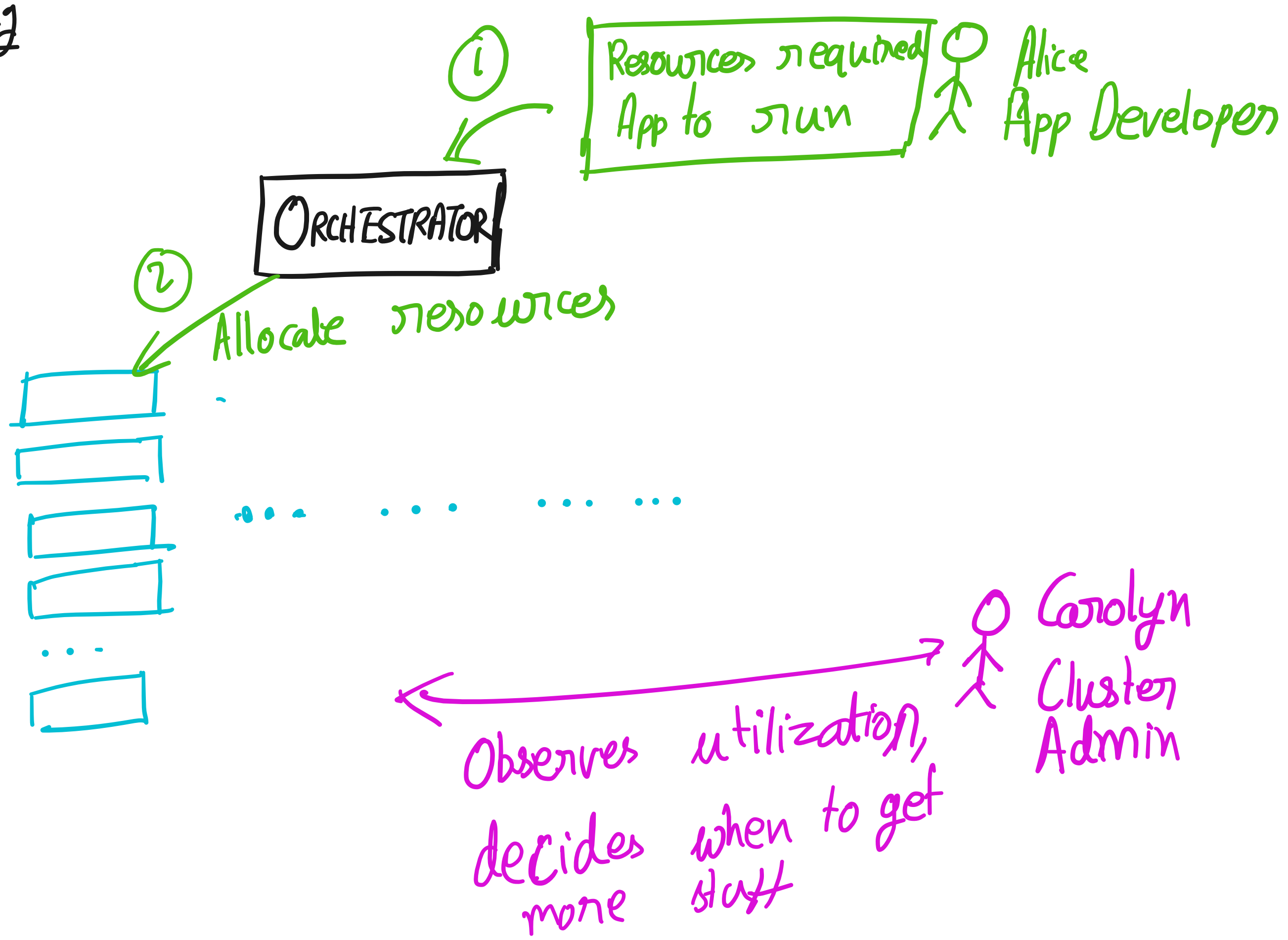
- Check decisions are correctly implemented

  ↳ Report bug if not

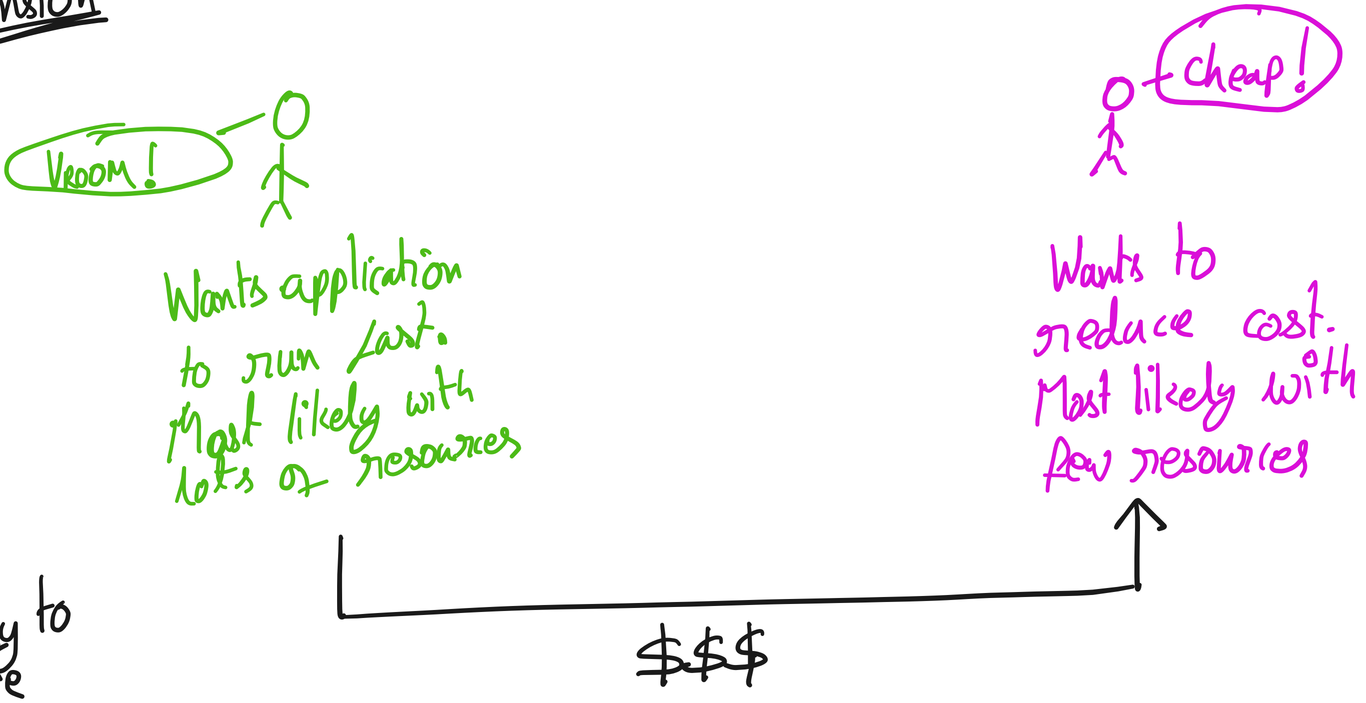
But, is this enough to be 'confident' about control loops?

o o o o o o

Auto scaling



Tension



One way to solve

But how much to charge & 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%

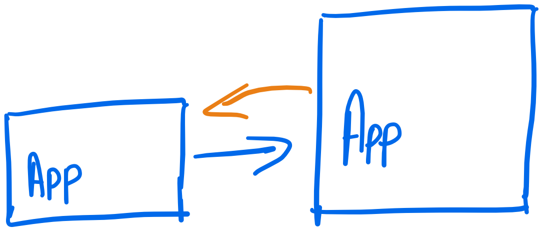
Quickly leads to the idea of autoscaling

Allocate & deallocate resources on demand.

Can be more efficient.™

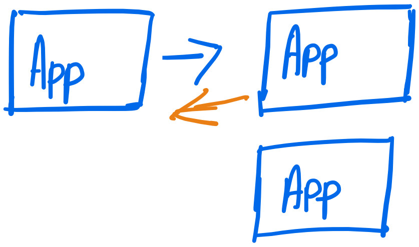
Like many obvious ideas, this one is very hard in practice

Vertical Scaling



- More CPU/
memory/
disk space/
...

Horizontal Scaling



- More copies — somewhere.

Some challenges

- Effects take a while to become visible

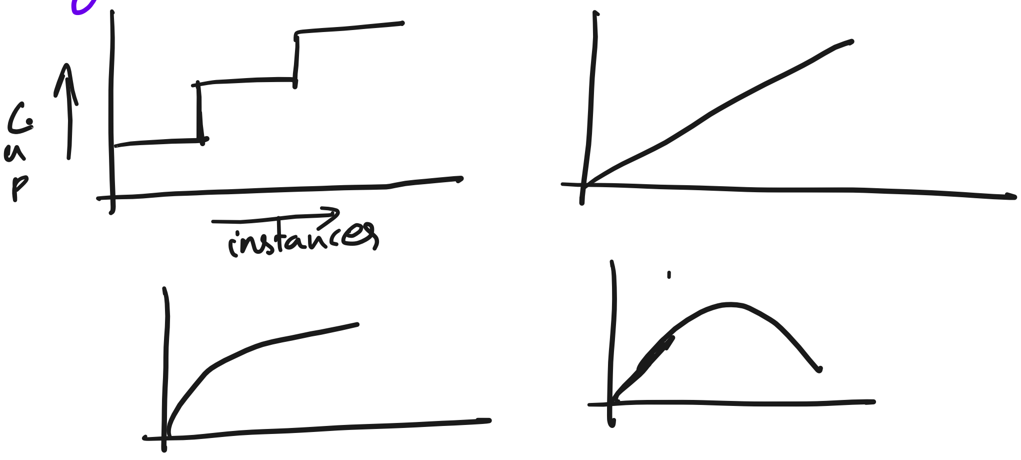
  Horizontal: Initialization

  Vertical: Kind of like initialization

Don't wait too long to scale !!

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

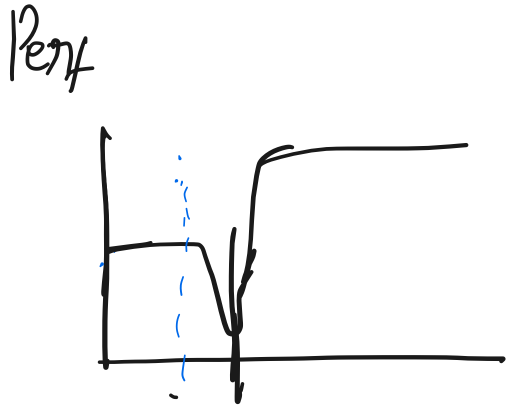
Steady state capacity

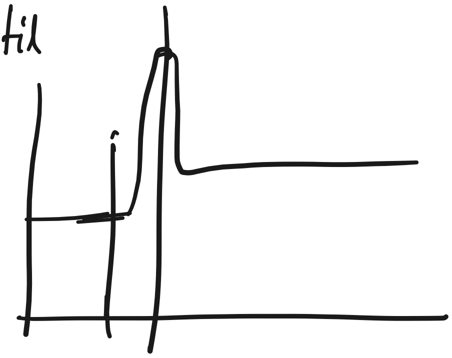


...

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

Careful about who to scale.

  - Databases

- Vertical scaling may not be safe

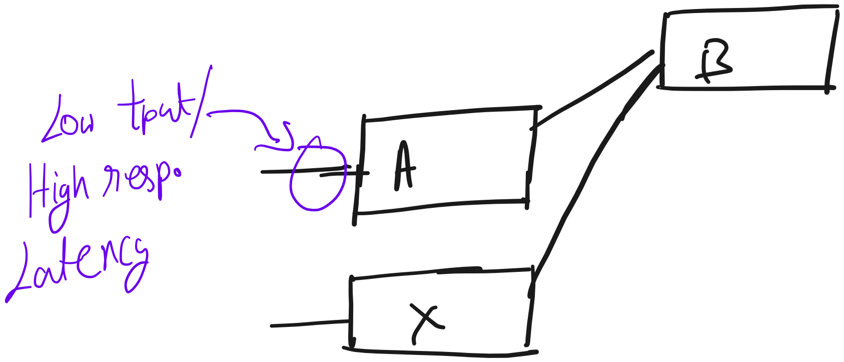
Autoscaling is too hard: Let us have someone else do it.

- Control loop

Q: Inputs

↳ Must be careful about when to measure

→ Must be careful about what to measure



→ Must be careful about how to measure.

↳ Averaging is the big hammer everyone uses here: gets 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

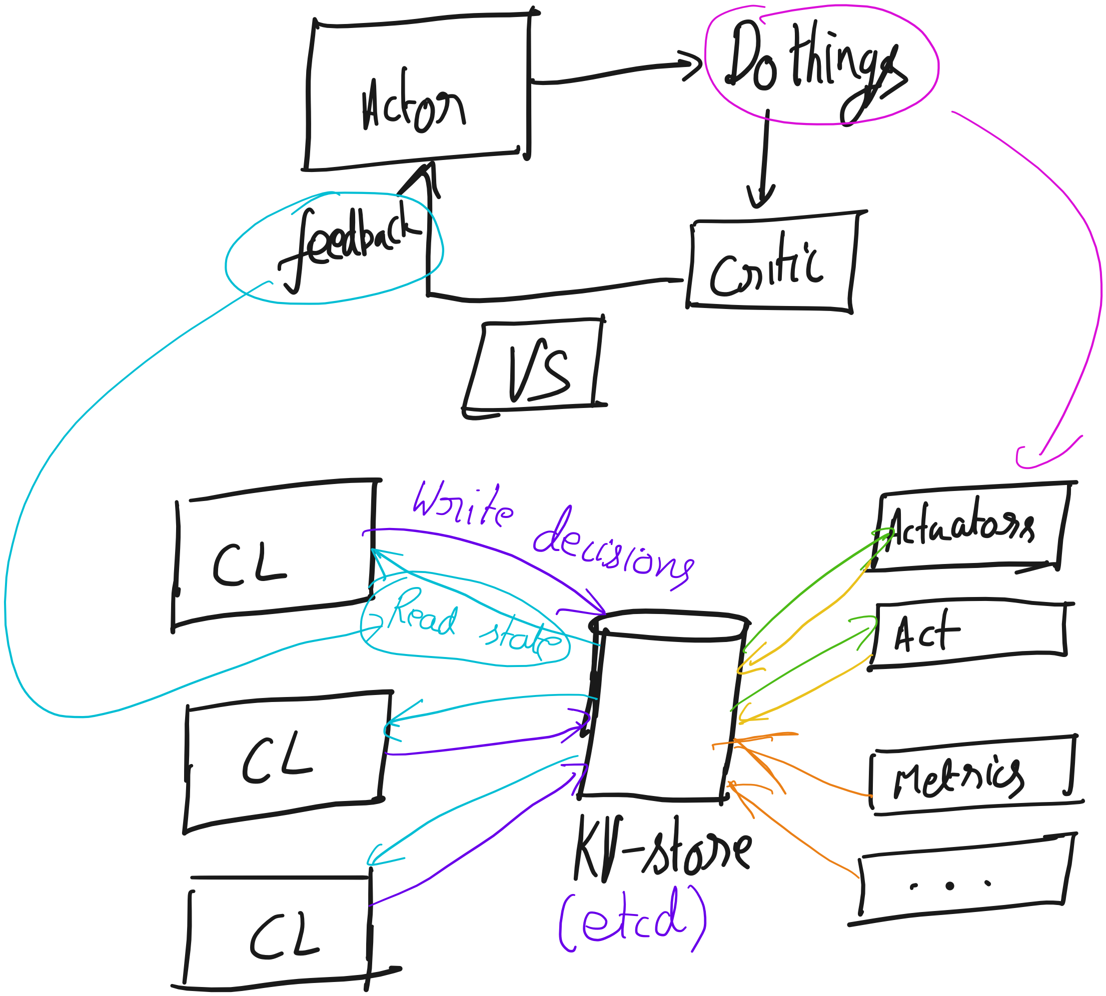
What one can?

SVM+ RL vs Control Loops (FIRM)

- First: Not a real distinction

Ben Recht & Others: RL is just fancy control theory

↳ Use feedback (critic) to decide whether action was appropriate



So what is different

- → More freedom on how to interpret/weigh signals
- - More freedom on chosing actions
- → Weights & actions can change due to feedback
  - ↳ Sort of a control loop for a control loop.

What is with the SVM bit?

- Two step

two step

↳ ① Use SVM to pick target for scaling

② Use RL to autoscale target

Why split?

→ Space of possible choices to consider

DS2: Does application knowledge help

Unsurprising: Yes

Surprising: Tricky to model & capture information

↳ Requires several assumptions beyond application knowledge.