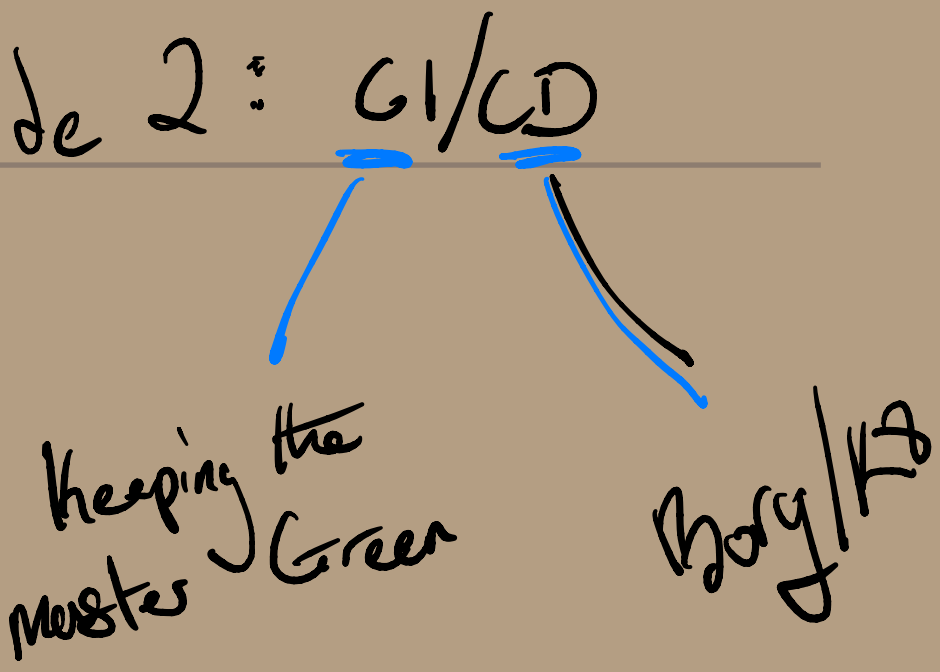
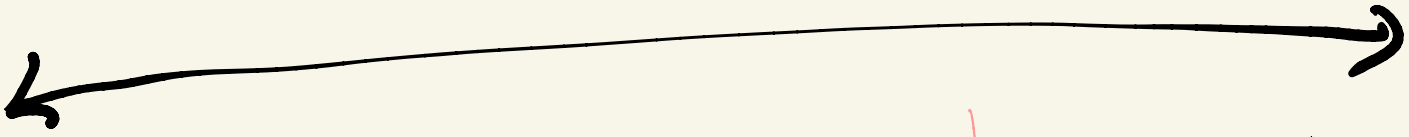
Episode 2: CI/CD



Today

1. Paper reading / analysis
2. CI / CD
   a. polyglot (pro/con)
   b. monorepo / polyrepo (pro/con)
3. Merge Queue (shopify) #1 / Submit Queue (Uber) #2
4. Borg #1 / K8 (Kubernetes) #2



Borg

Design choices
* understanding benefits & tradeoffs
* properties of systems (reliability etc)

·System design

Submit Queue

* Design choices
* Algorithms for conflicts / speculation

algorithmic design

How do properties of system's characteristics/ workload dictate constraints

Paper structure

"How this paper advances the field"

⭐ Related Work

"the mostly related work..."

(1) prior work in the space & how they tie to the current paper

(a) solve same problem with diff technique

(b) similar techniques for different problems

(c) same domain

Eval / Results

Hidden Assumptions

experiment setup

* How much better than related work

metric of success

(a) throughput : RPS
(b) reliability : mean time to Recovery

Workload

(a) inputs
(b) services
(c) Infrastructure

under what conditions will this system be better

This is always BETTER !!!!!

Discussion

* Open challenges
* Limitations

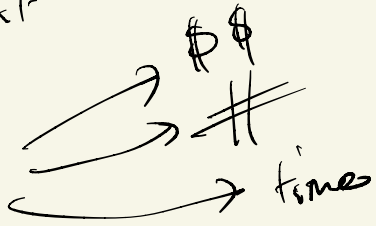
* look across papers & identify limitations
  (a) What is the glue to put the papers together
  (b) Do they all make the same assumptions? If NO!!
      How to make the same assumption (s)

Introduction

* Background terms / problem context
* highlevel architecture
* highlevel benefits

Good Introduction

(1) Background / problem context
   (a) what is the problem
   (b) why is it important



(2) why is this challenging
   (a) why is the community struggling

(3) highlevel of the system / solution
   (a) key insights / observation (magic) underlying the system

(4) summary of eval highlight (marketing)

Architecture / Design / System

① what is the system?
what are the components?
How do they interact w/ eachother?
what is a summary of their features

Hidden Assumption ⇒ How do they impact the design?

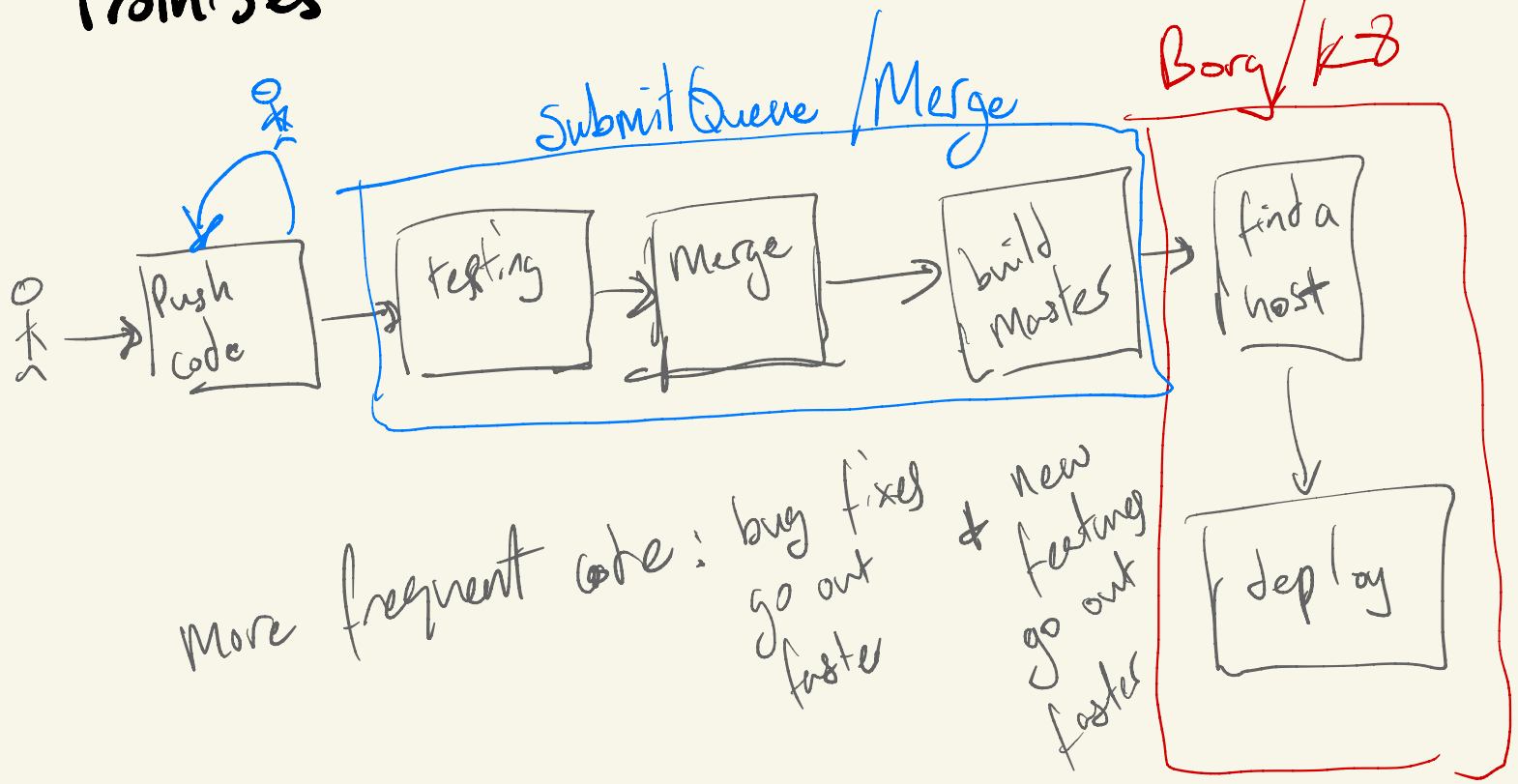
if the H/W or service/app or limiting factor changes

Citations

- Top Venues (conf/workshop) ✓
- "closest ..."
- ACM Digital library
- Group that dominates then go their website

CI/CD

Promises Microservice ⇒ Move fast



more frequent code: bug fixes go out faster & new features go out faster

100 - 1000 code changes per day

100 per week
t per week

Config is code

polyglot

write in any language

advantage(s)

* multiple library echo system
  ↳ use "best" language for task
* use your "best" language
* choose language/runtime based on task properties

disadvantage(s)

* complexity for interactions across boundaries (RPC)
* Harder to transfer knowledge
  - libraries
  - people
  - diagnosis

monoglot

every one in the org writes code in one language

advantage(s)

* library/code reuse
* Deep expertise in a language
* transparency across organization
* impact on the language community's trajectory

disadvantages

* bootstrap a lot of libraries
* restrict developer pool

Poly repo

one repo per team

* Amazon
* lyft
* Netflix

Advantage

(*) independence → each team is free

Disadvantage

(*) harder to share things

(*) harder to enforce global practices

monorepo*

one repo to rule the org

* Face Book
* Twitter
* Google
* Uber
* Black Car

Disadvantage

* large code ⇒ download/compile
* keep master green

advantages

* "culture"
  - code style
  - testing practice

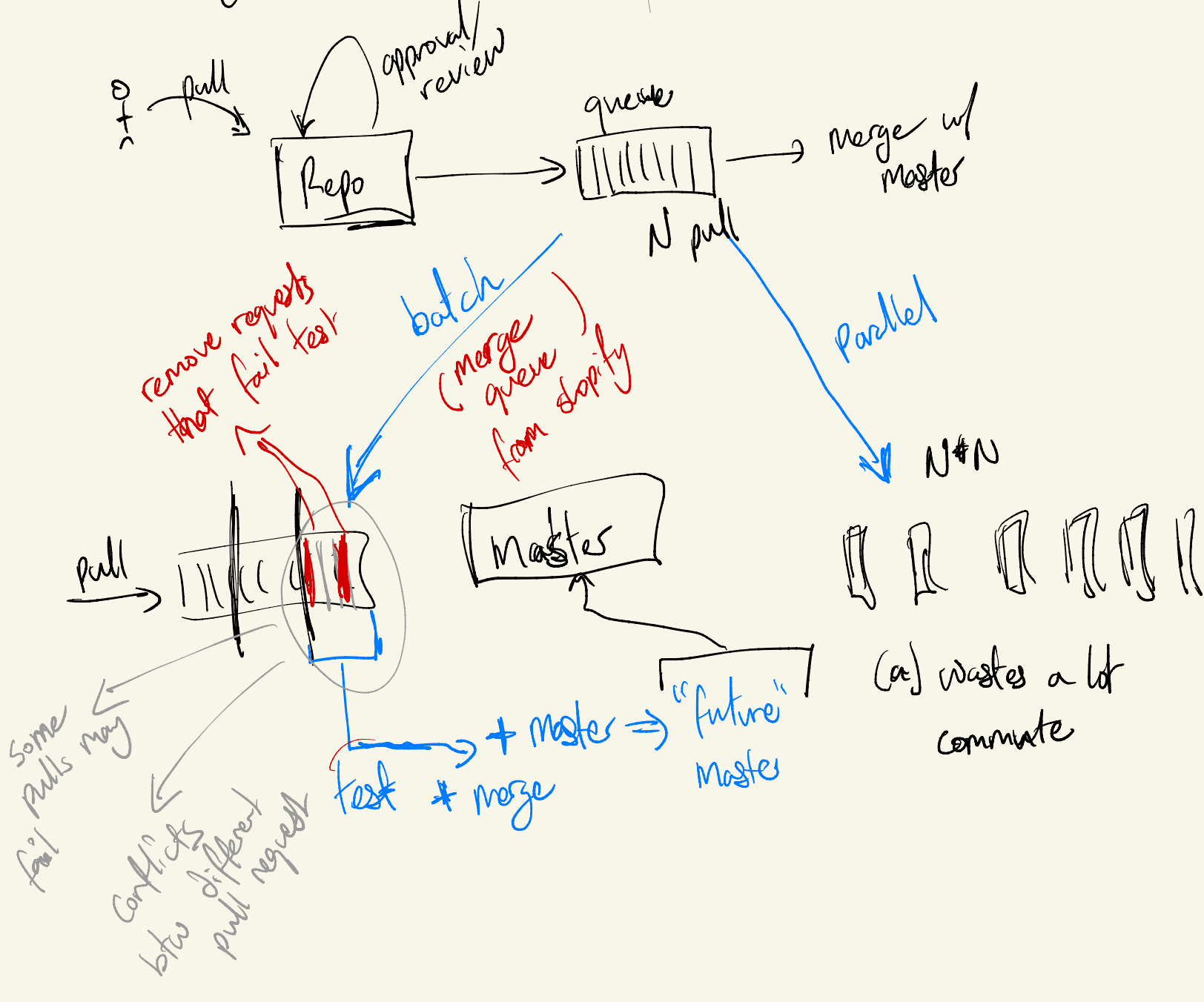
Submit Queue

Benefits of a green master?

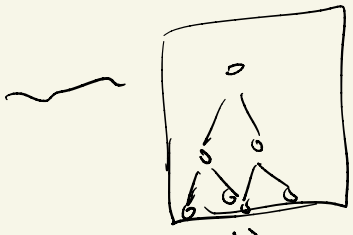
(1) takes time to fix a red master

(2) higher developer productivity (you code wouldn't be overwritten by rollbacks)

(3) quickly rollout bug fix



Submit Queue



speculation tree

assign weights

P(succ)

func (developer name)

Business

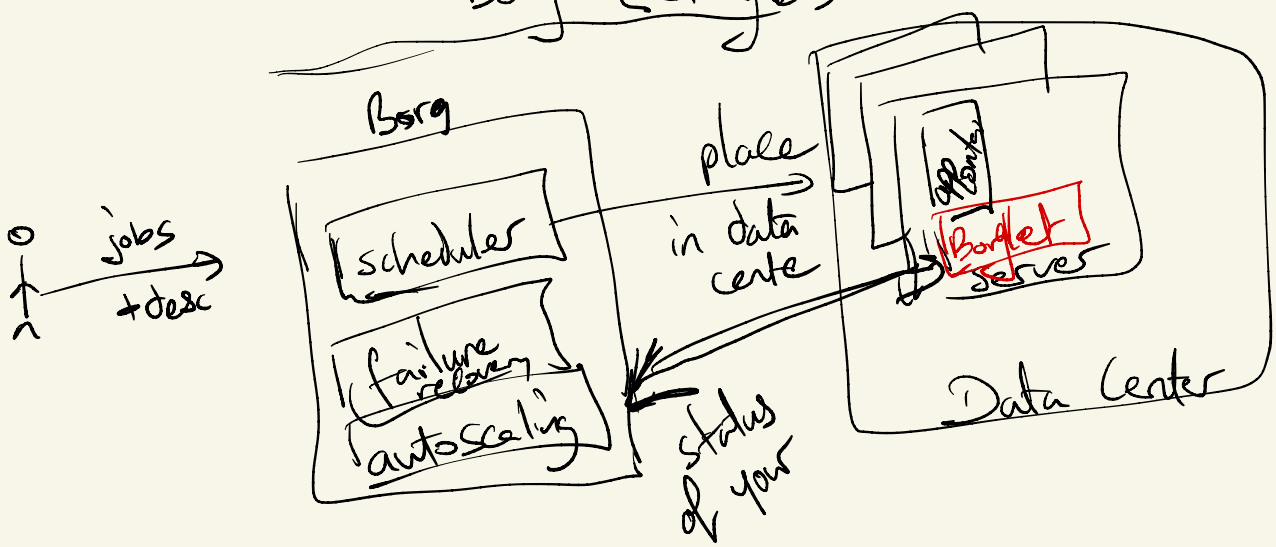
priorities

identify conflicts (tree into graph)

conflict

* build knows which files/targets get impacted by a change

Borg (Google's container orchestration)



Borg manages your cluster

Debug

efficiency

place jobs in a fashion maximize server util

fault tolerance

monitor & ensure that service is always

admission control

* priorities

background (random)

foreground (user facing)

Scheduling: How/where to place services/jobs

(1) high util

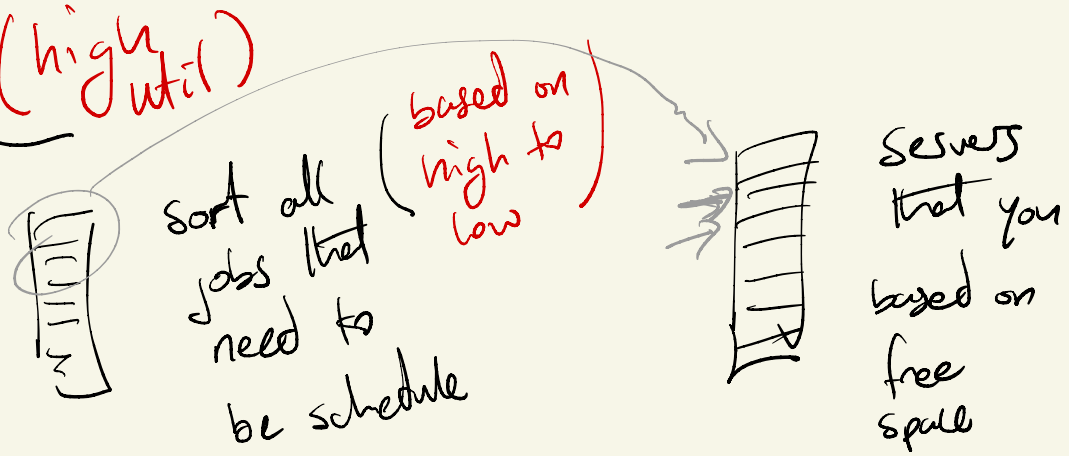
*(2) constraints in term of placement (EU)

(3) fast

(4) fair ← Priority (foreground/Background)

↳ $$

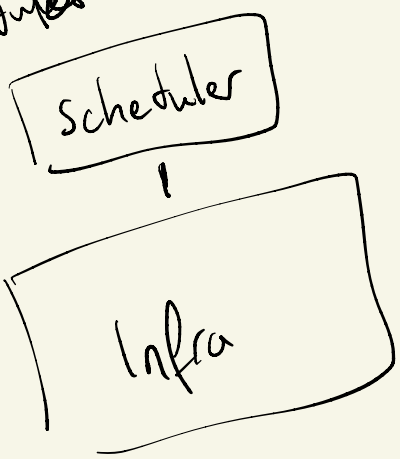
Bin Packing (high util)



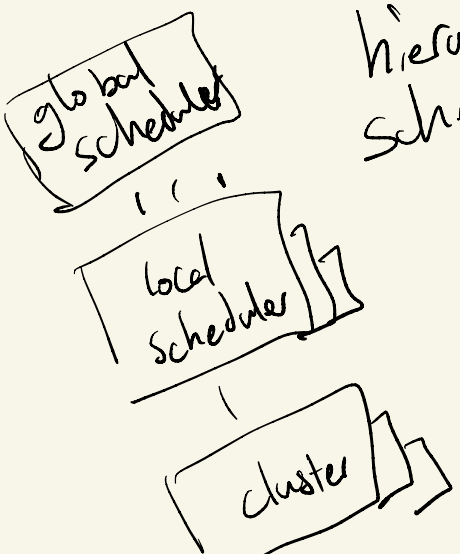
1 Rack of servers

Rack of server

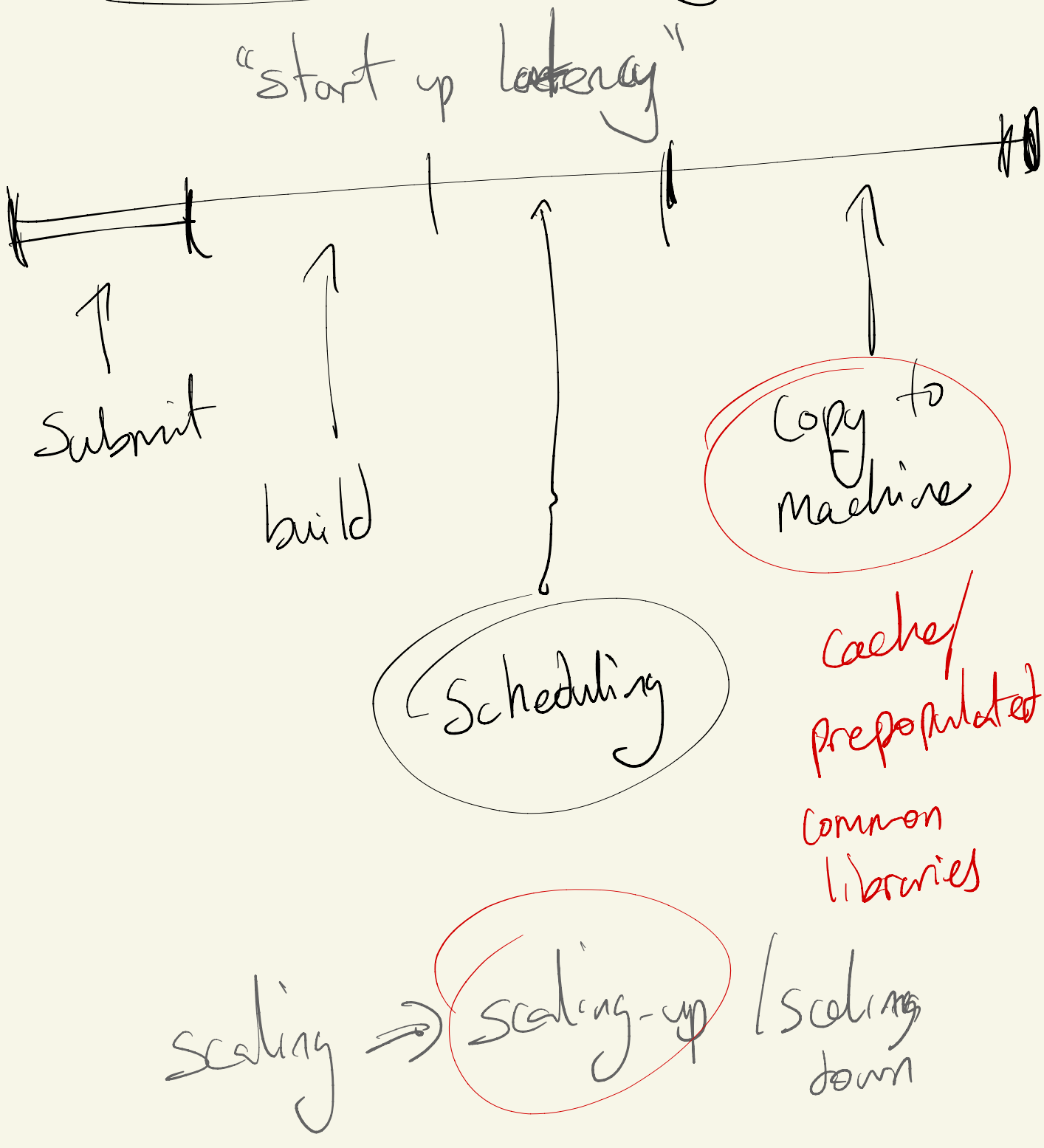
one scheduler



hierarch scheduler



Speed of Deployment



scaling ⇒ scaling-up / scaling down

Today

* Paper reading / writing
* CI / CD
  * polyglot / polyrepo
  * Submit Queue / MasterQueue
  * Bory: high level components
    * Scheduling