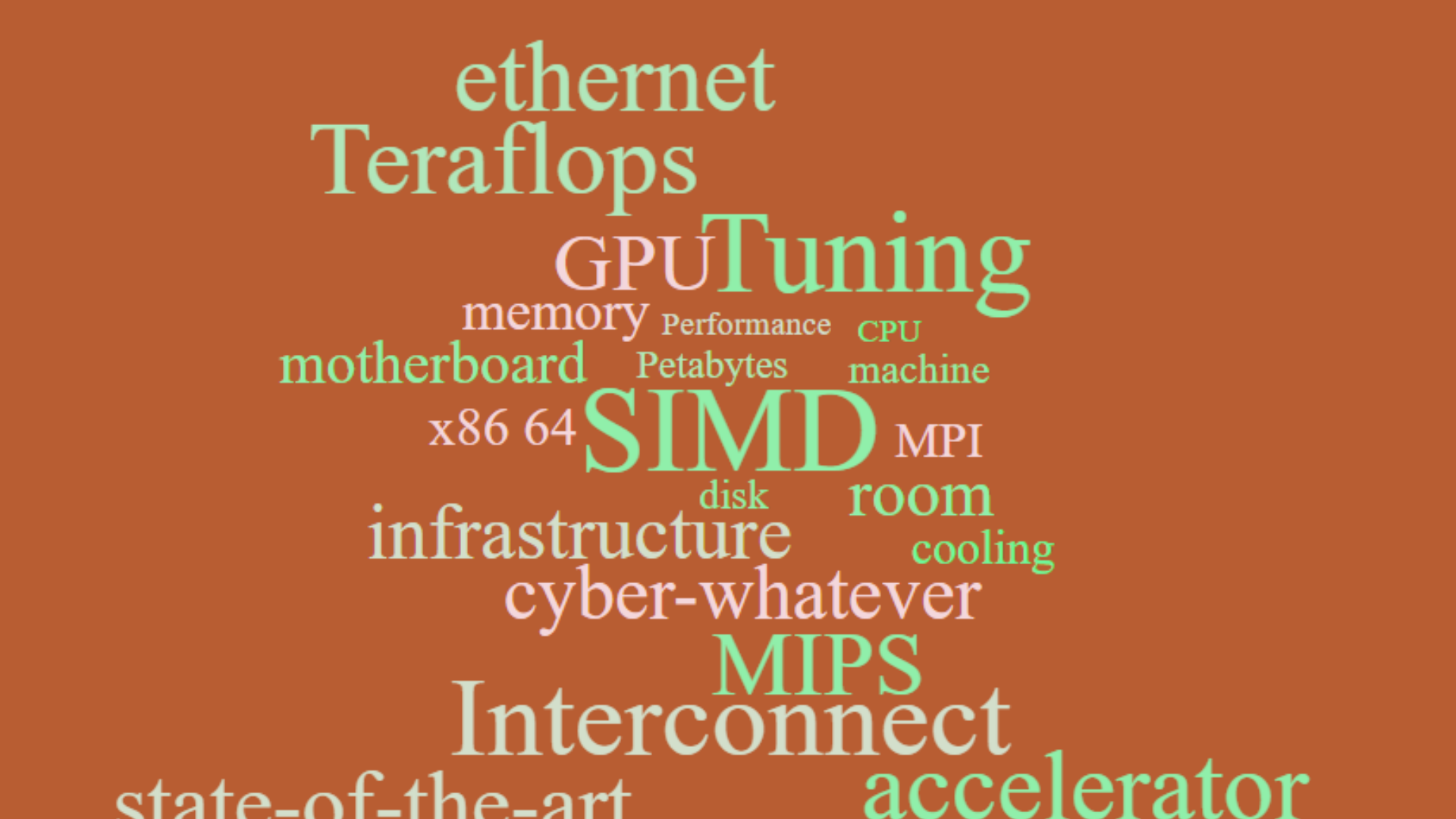
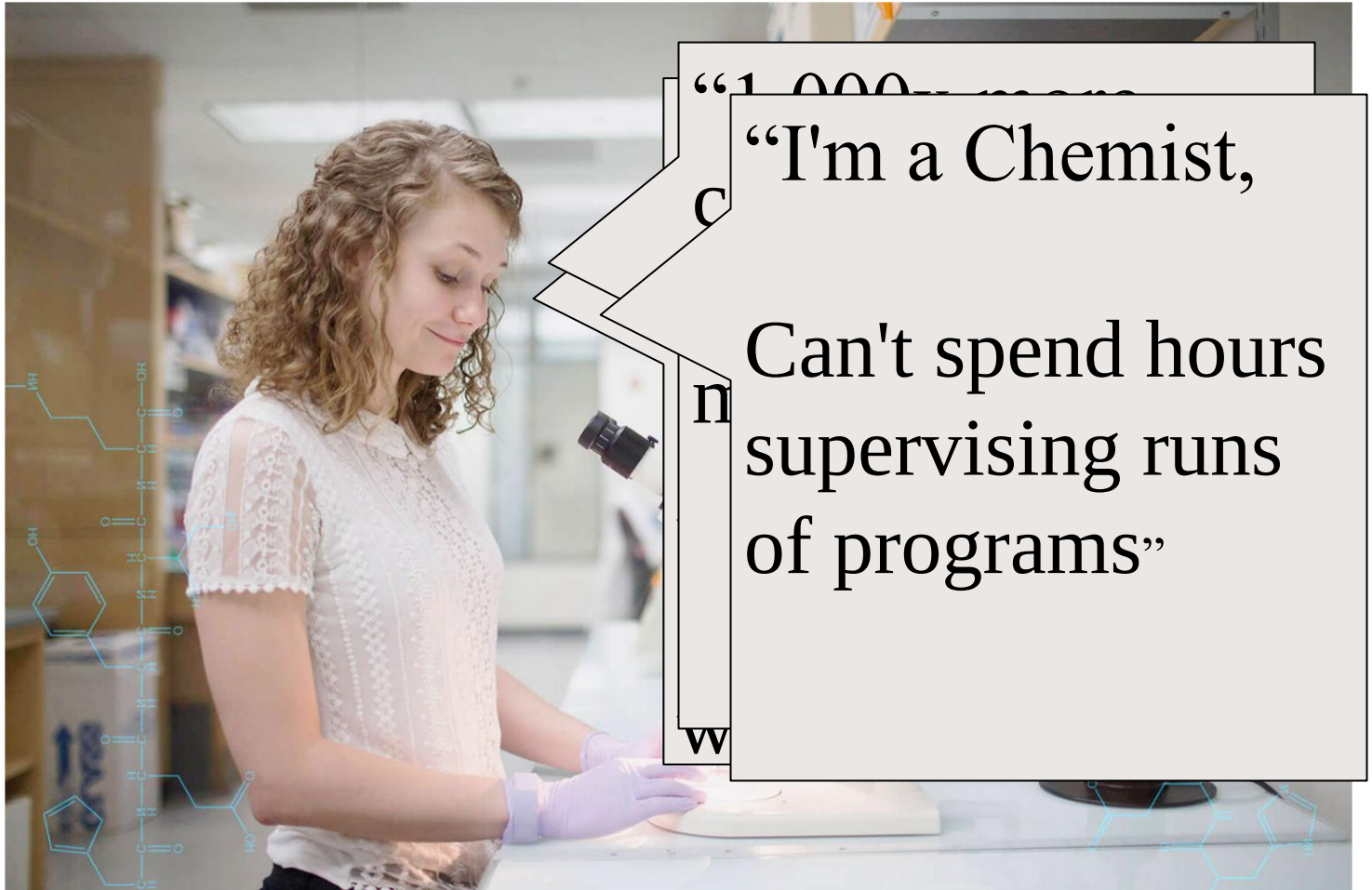
Philosophy of High Throughput Computing

Greg Thain


ethernet
Teraflops
GPU
Tuning
memory
Performance
CPU
motherboard
Petabytes
machine
x86 64
SIMD
MPI
disk
room
infrastructure
cooling
cyber-whatever
MIPS
Interconnect
state-of-the-art
accelerator

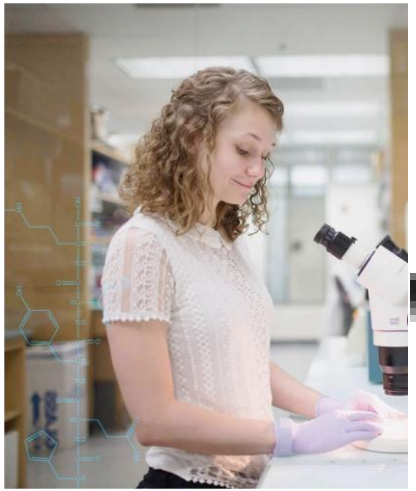

Start with People


People have Problems


“I'm a Chemist,
Can't spend hours supervising runs of programs”

The Mythical Golden Laptop

The Mythical Golden Laptop




This machine is a...
Do Not Power ...
... Down!!

ORGANISATION EUROPÉENNE POUR LA RECHERCHE

CERN EUROPEAN ORGANIZATION FOR NUCLEAR

1211 GENÈVE 23 (SUISSE)

This machine is a ser

DO NOT POWE

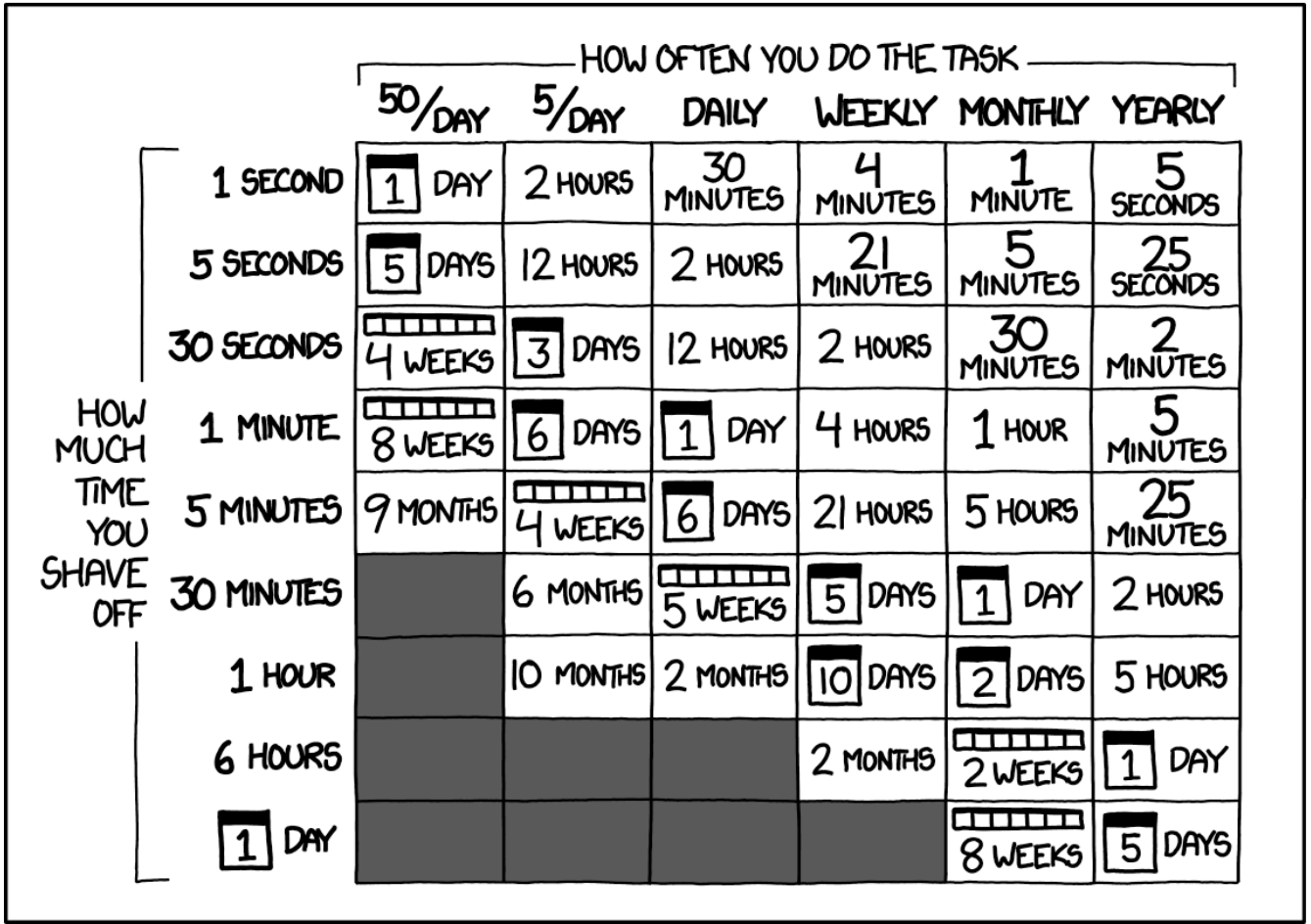
DOWN!!

Laziness: The 1st virtue

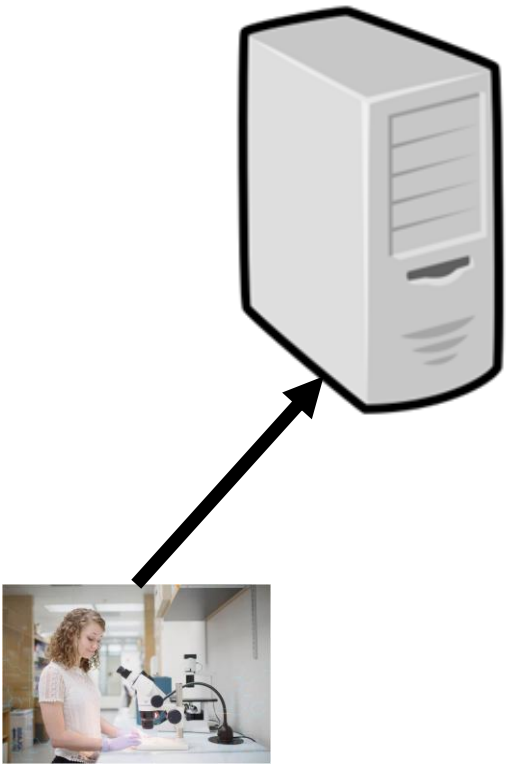
(xkcd: 1205)

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?

(ACROSS FIVE YEARS)



The HTCondor Access Point(AP)



Software to manage all work

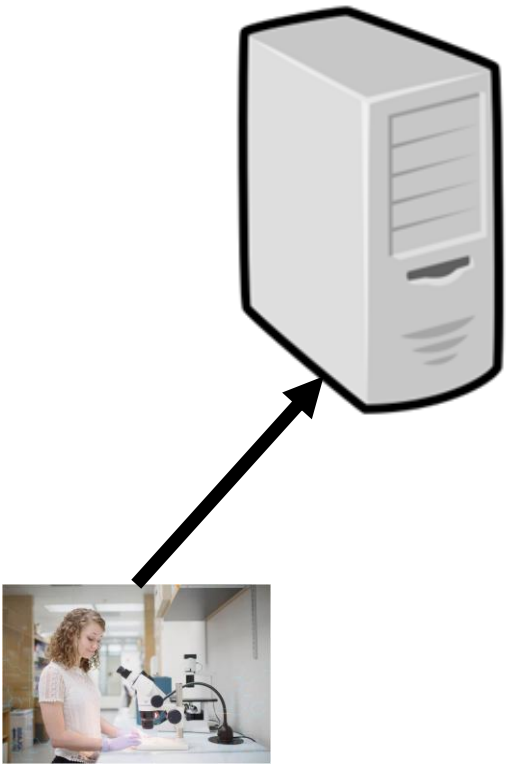
All needs declared

Reliable – like money in bank

Implements user's policies

"Submit it and forget it"

The HTCondor Access Point(AP)



Software to manage all work

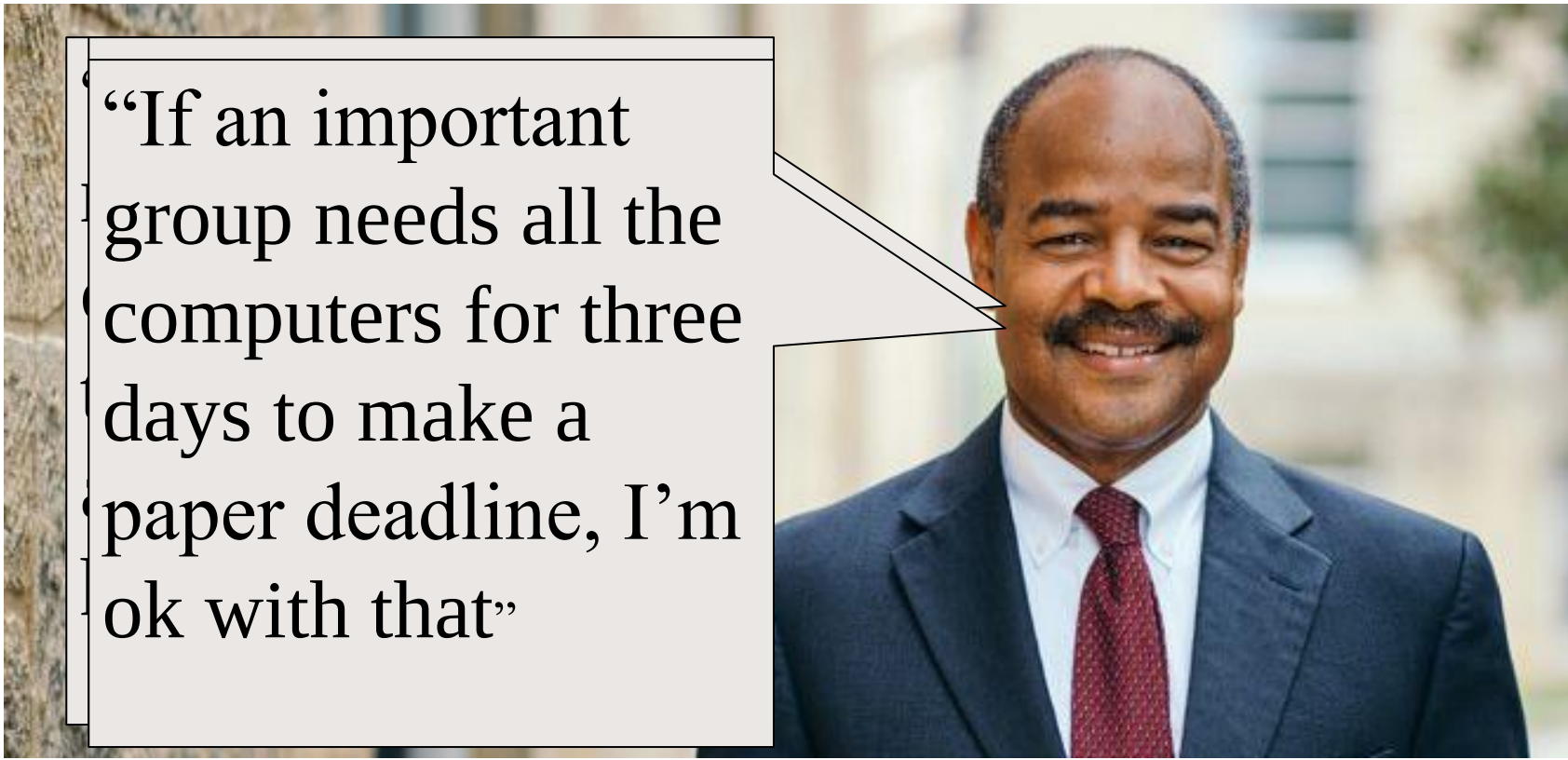
All needs declared

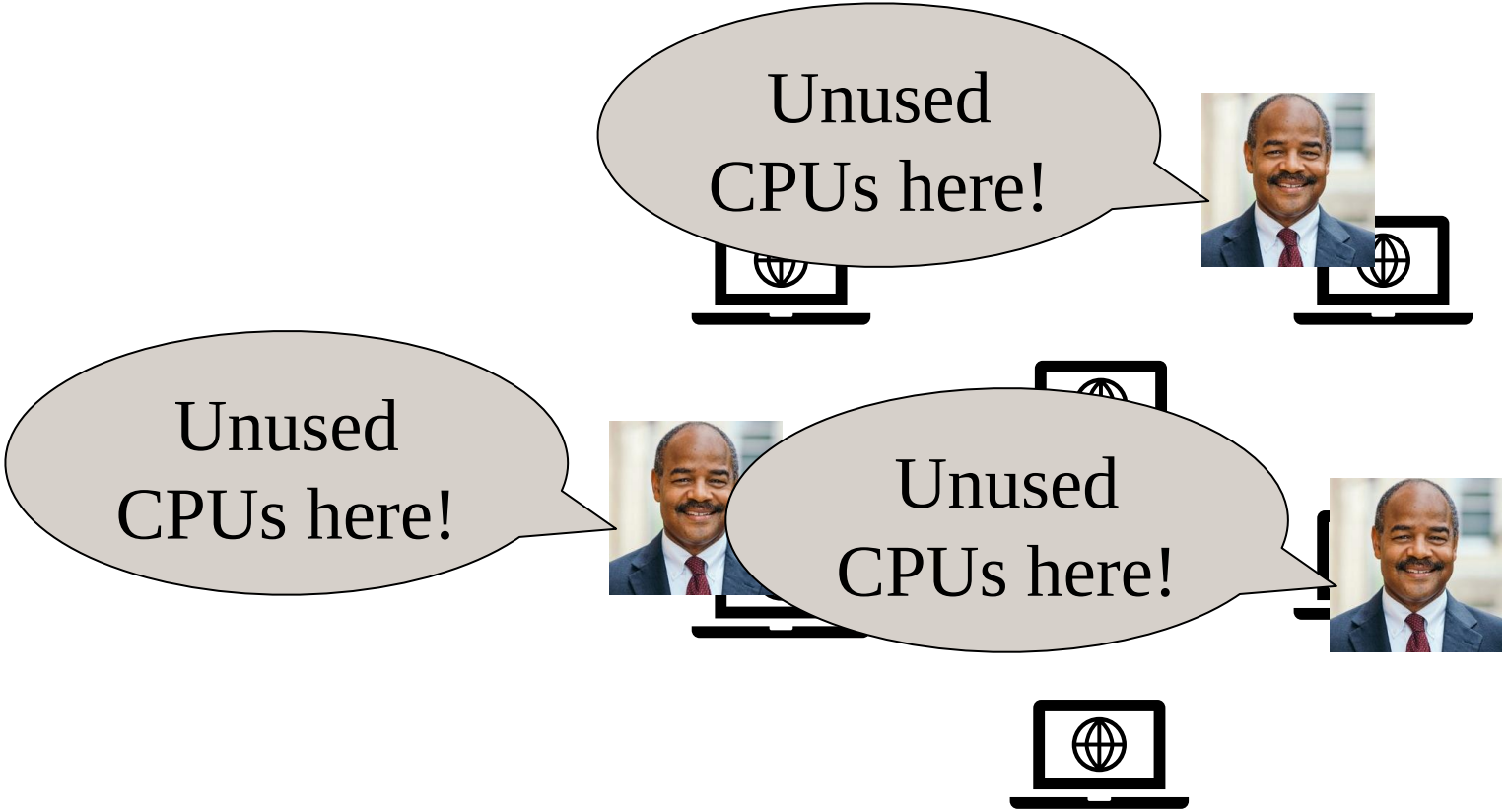
Reliable – ~~like money in bank~~

(like money in mattress?)

Implements user's policies

"Submit it and forget it"


“If an important group needs all the computers for three days to make a paper deadline, I’m ok with that”


Unused
CPUs here!
Unused
CPUs here!
Unused
CPUs here!

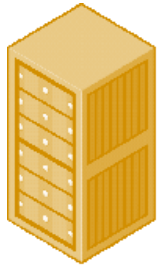
The HTCondor Execution Point

Where work runs

Also has policy

Policy is declarative

Probably many of these



on many machines

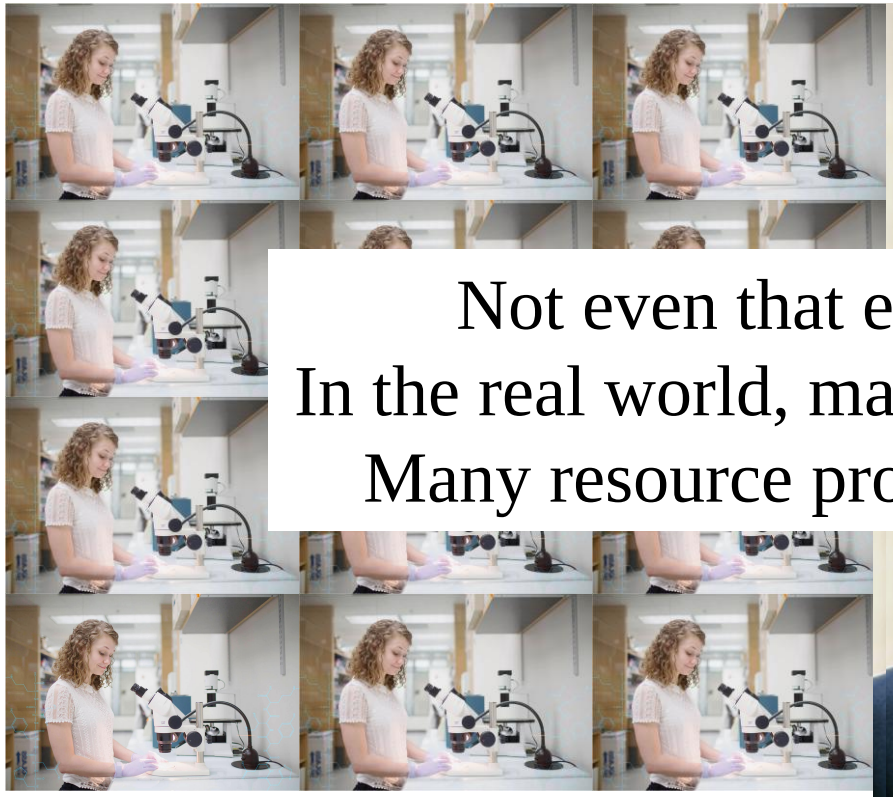
Implies machines are heterogeneous

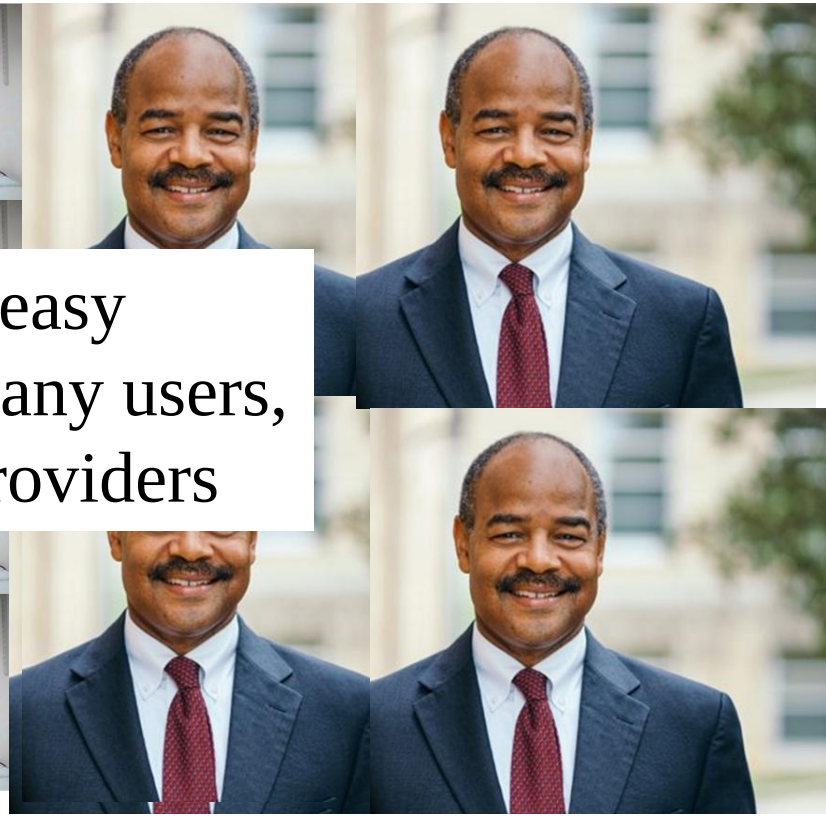
Could be machines owned by others

Could have different polices

Could be places without shared filesystem


HTCondor
Manages
These
constraints





Not even that easy
In the real world, many users,
Many resource providers

Two-faced nature of HTCondor

Split responsibility:

Worker side

AP side

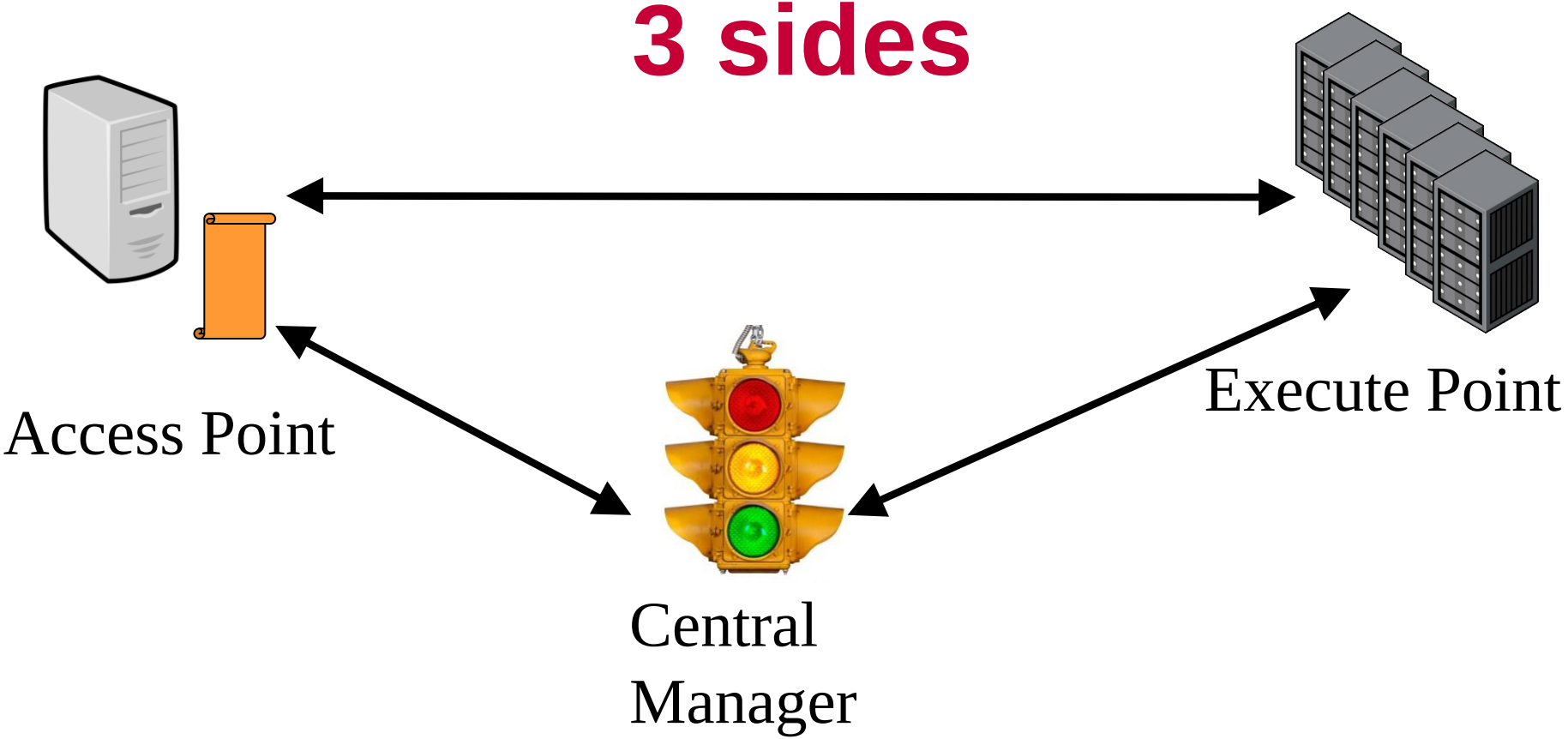
We ***encourage*** different policy on both sides

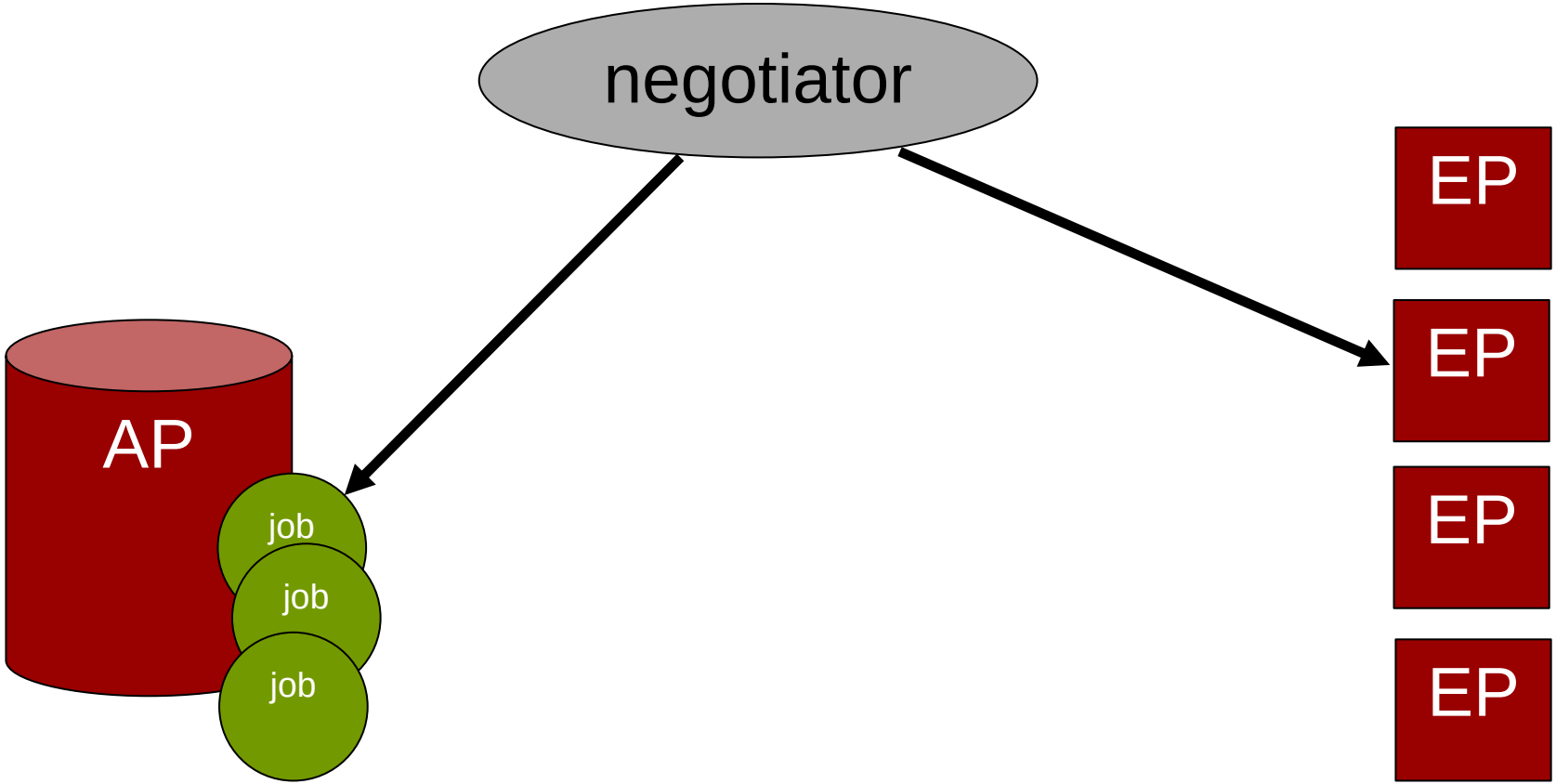
(who wins in a collision?)

Always focusing on responsibility of the side

Always consider where responsibility goes

Overview of condor: 3 sides




negotiator
EP
EP
EP
EP
AP
job
job
job


DOUBLETREE
HOTEL

Useful to think about moving EPs to the work

Useful to think about assigning EPs to the work (temporarily)

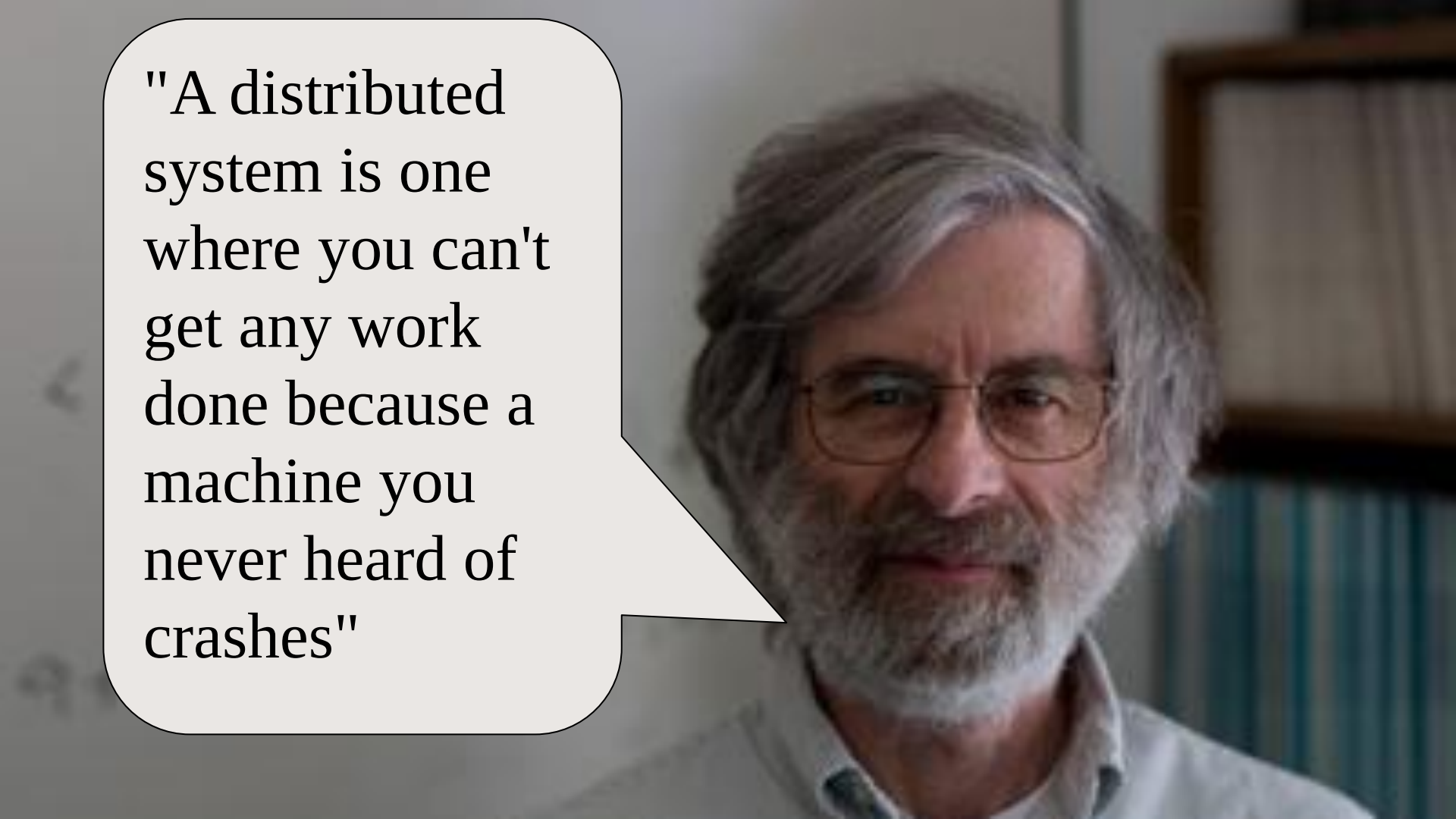

negotiator
AP
job
job
EP
EP
EP
EP
Not a guaranteed assignment!

This is a distributed problem.

Distributed because of **people**

Not because of **machines**.

Our goal is to satisfy all these constraints.


"A distributed system is one where you can't get any work done because a machine you never heard of crashes"

What we mean by "Distributed"

Distributed authority (people)

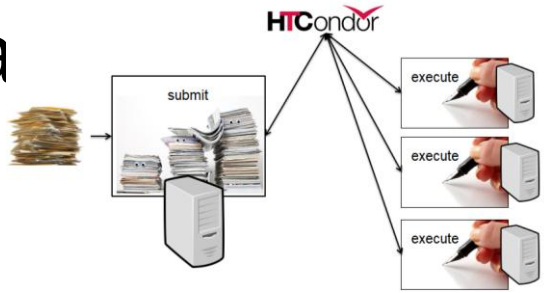
vs

Distributed machines – is k8s "distributed"?

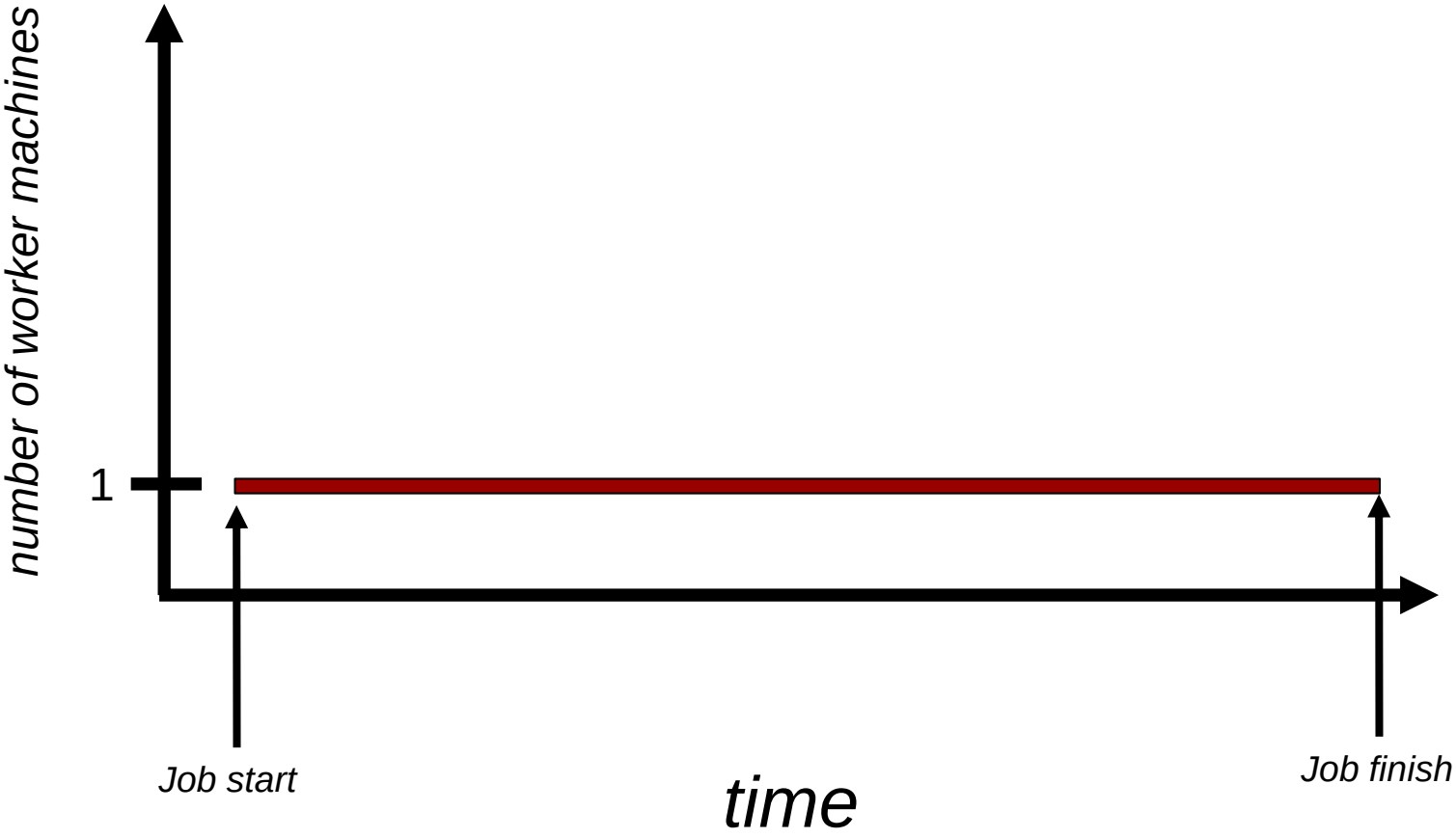
The Unstated Assumption

“All work” can be broken up into smaller jobs

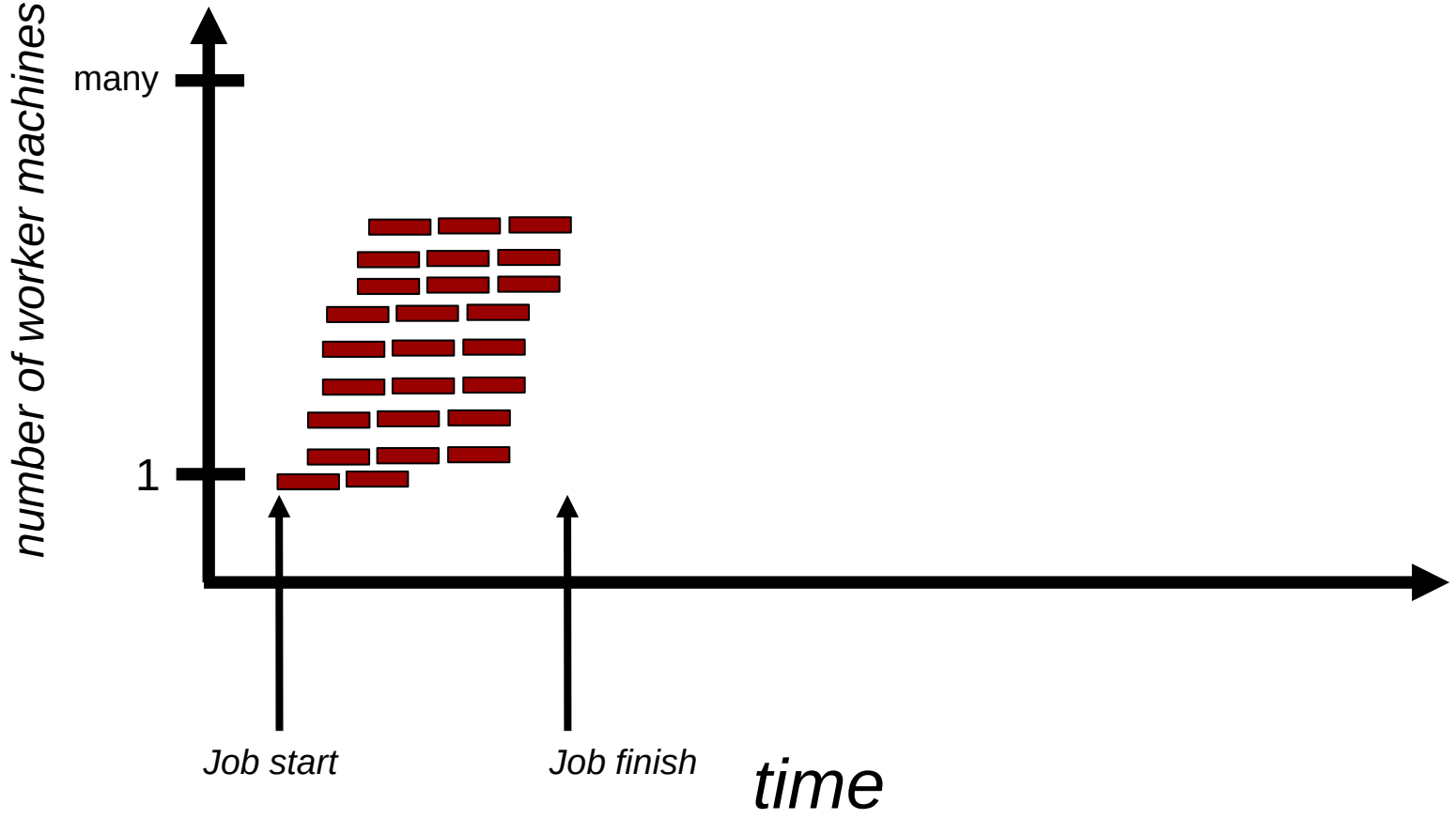
- Jobs (mostly) independent
- Smaller the better (up to a point)
- Files as ipc
- Any interdependencies via
- Optimize time-to-finish
  - not time-to-run



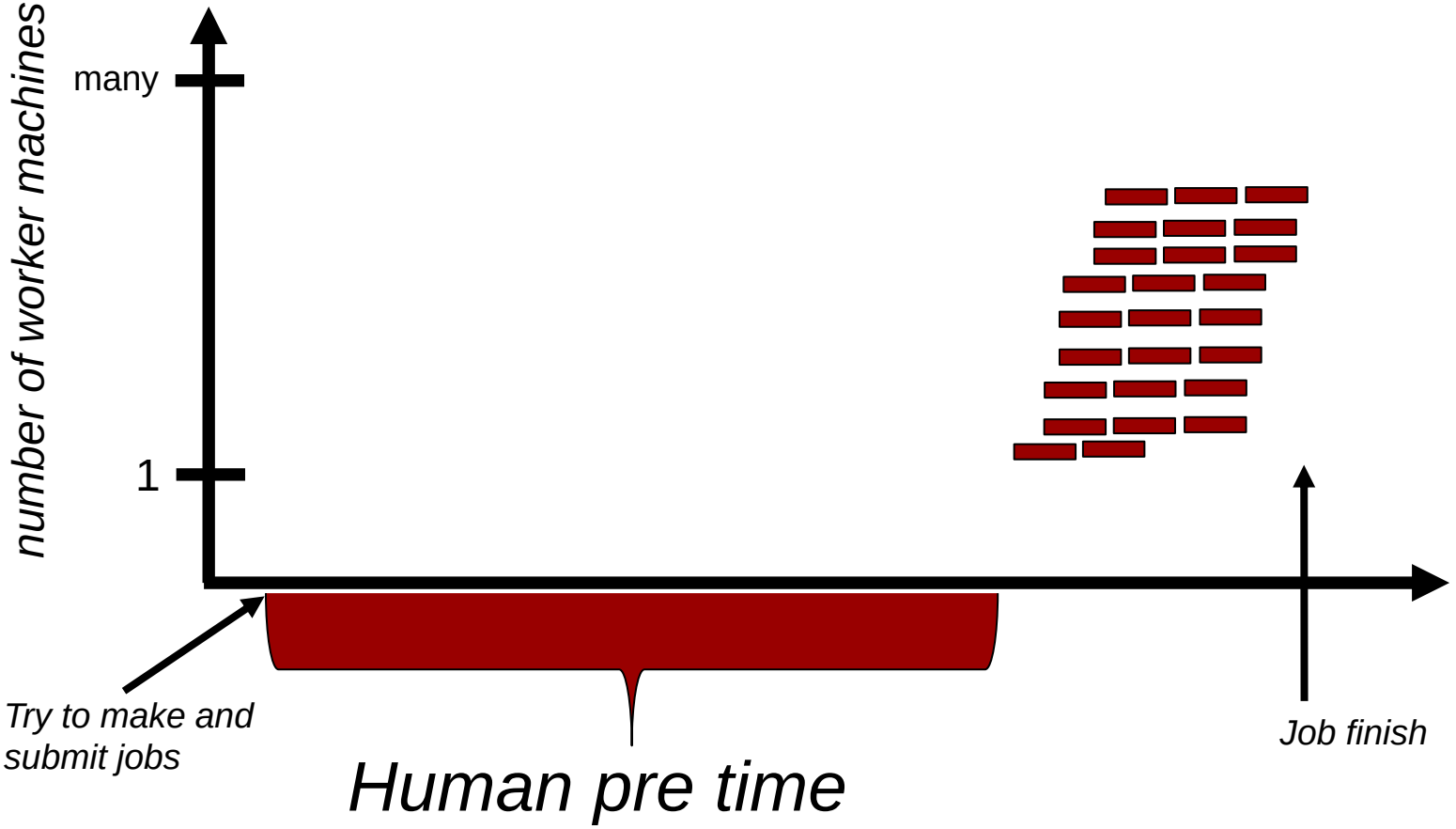
Go from …



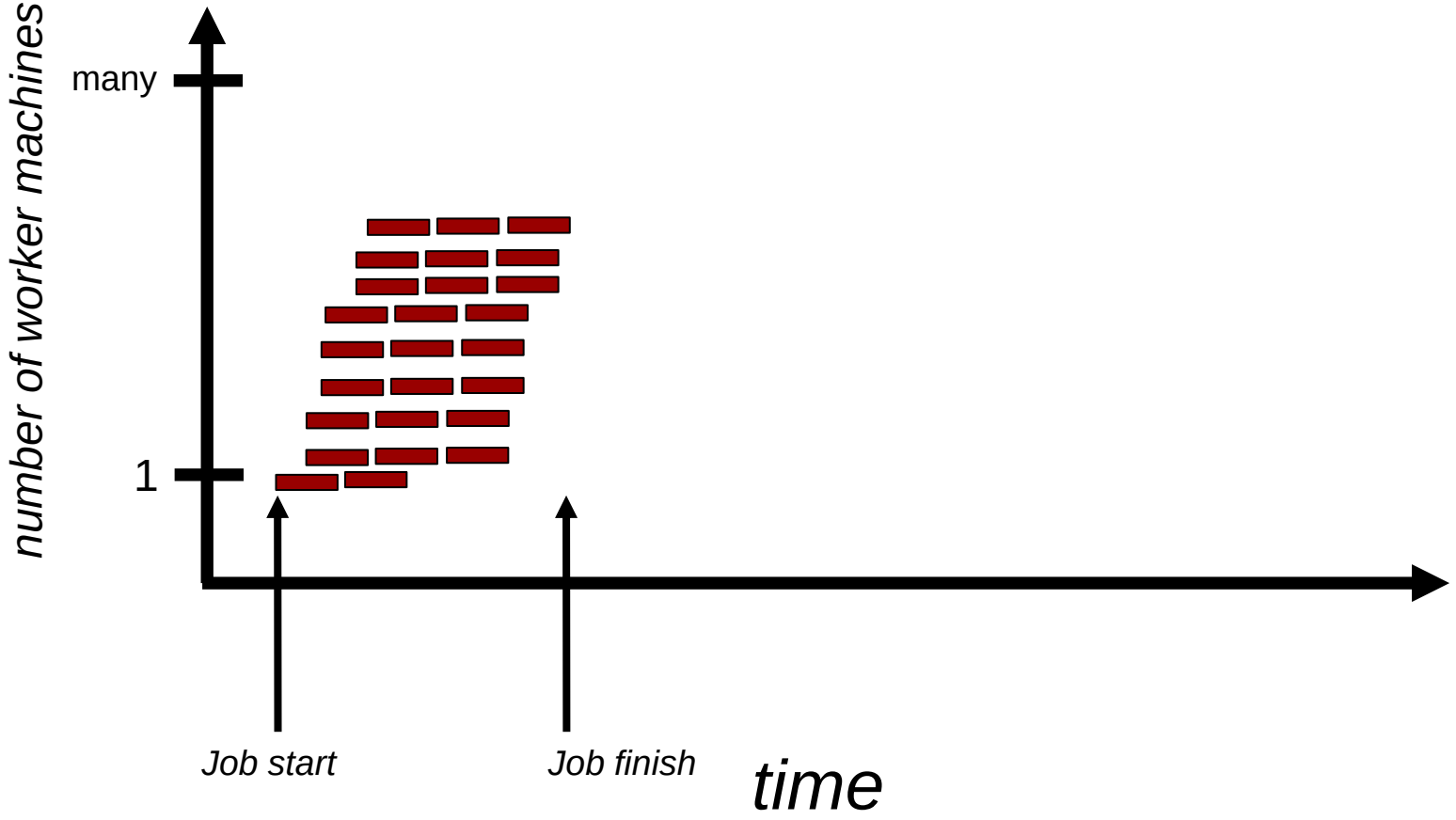
… To



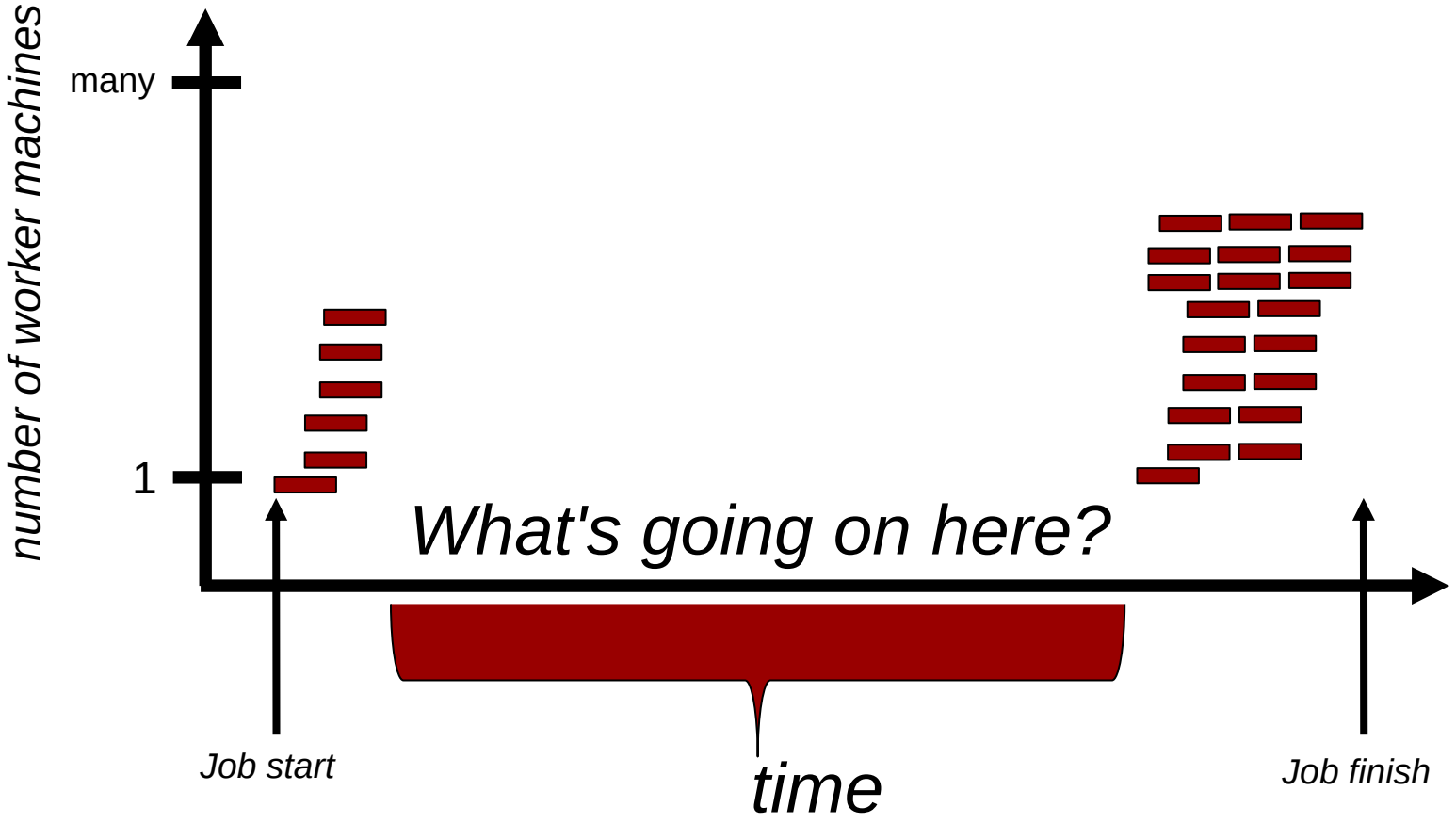
But what about this time?



Or what about?



this


number of worker machines
many
1
What's going on here?
Job start
time
Job finish

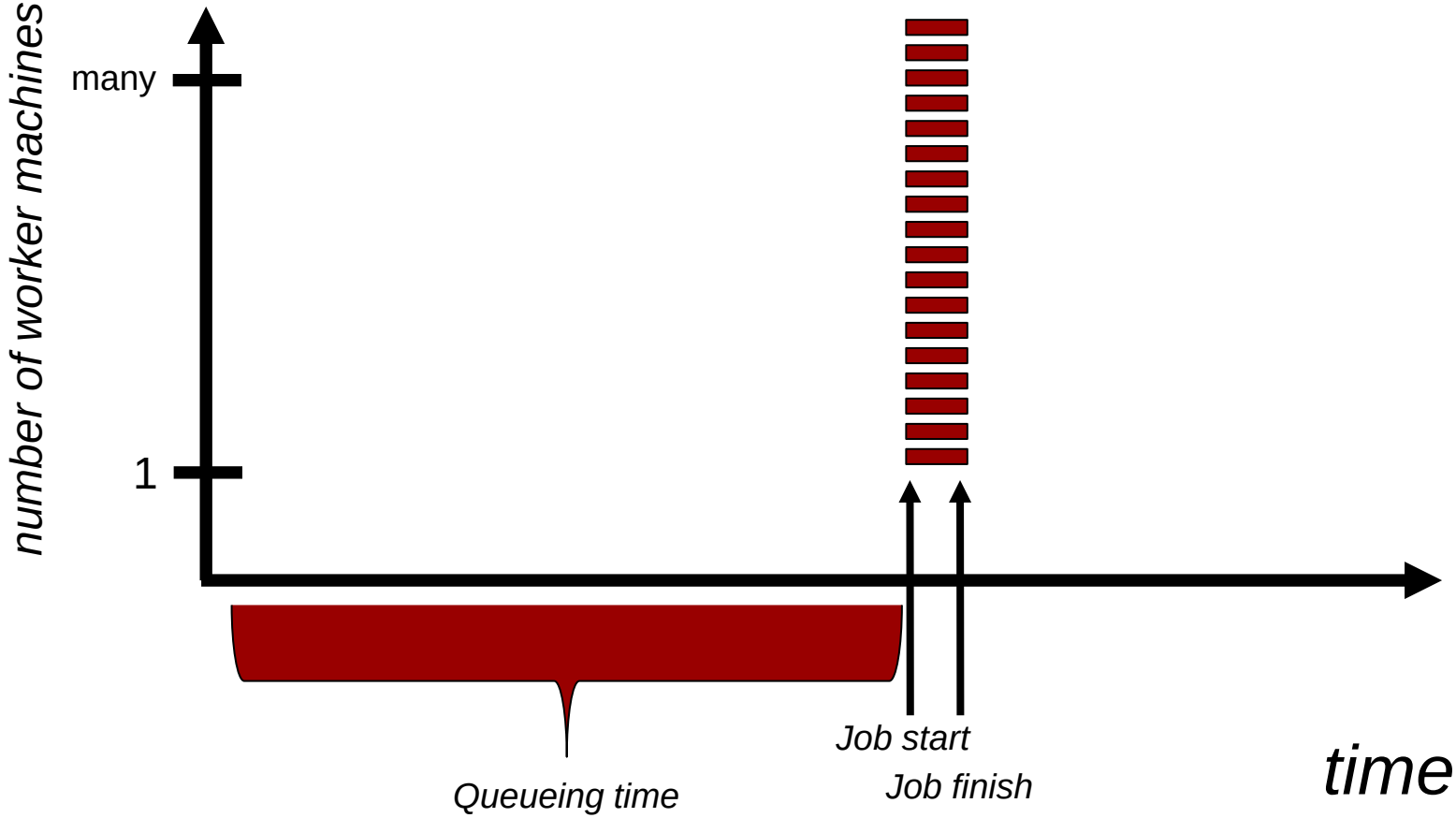
With the implication…

There's a LOT of jobs!

A lot of different jobs

Very different from HPC/MPI

The MPI / HPC Model




HTCondor

The Ideal Job

Ideal job, with "Job Nature" is pure function

$$\text{Output} = f(Input)$$

Completely described by submit file
No side effects
But we don't live in a perfect world

Things like an ideal Job?

Genetic Search

Monte-Carlo simulation

- (careful with random seeds)

Compression of large input

What is not ideal Job?

- Web server
- Database
- Interactive job

All while being reliable…

Reliability 1st priority

We can make HTCondor ***fast enough***

w/o sacrificing any reliability – no screw polishing

Scaling via many APs

Adding APs just helps scaling

Allows submission near the user

“Submit locally, run globally”

But … What about data

- Data is kind of "backwards"
- It lives at your home, and you want to send it everywhere
- Need "door" or "portal"

Just like leasing compute

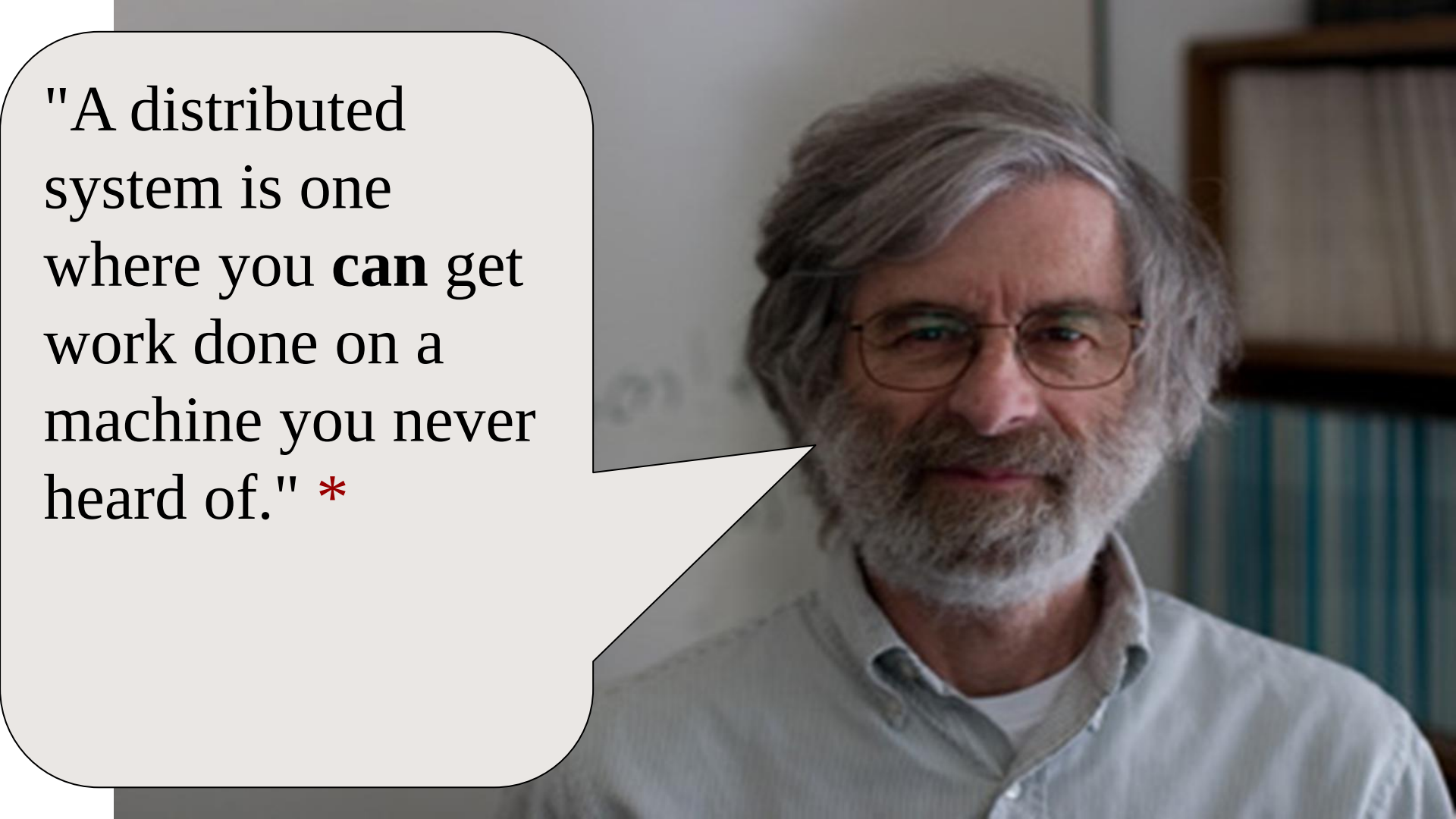
- A Pelican "cache" (or stage) can make a temporary home for your data
- Helps bring objects to your capacity

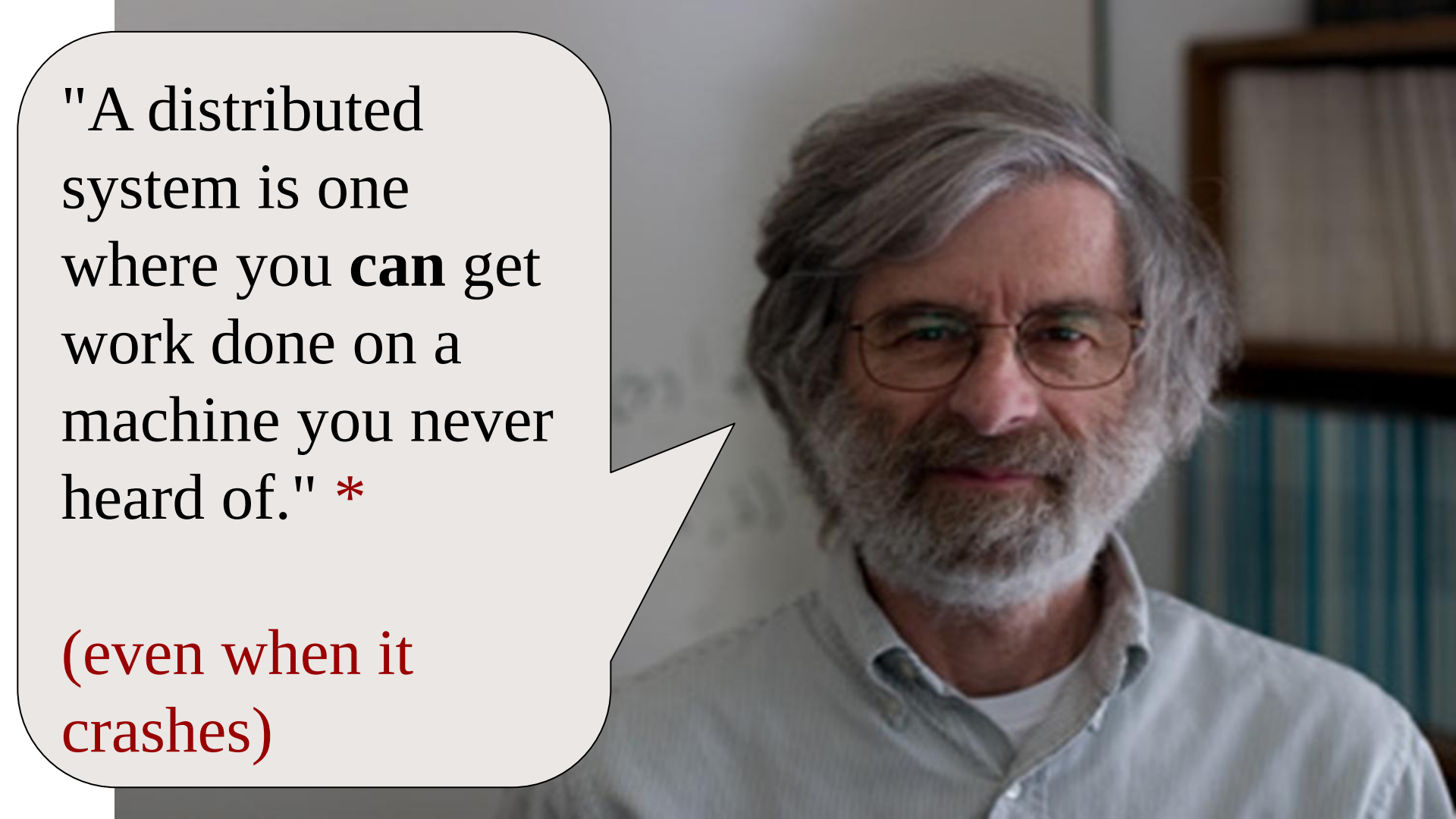
Putting things all together…

We do two major things to solve problems

1) We build open software to do HTC

2) We build open organizations to do HTC


"A distributed system is one where you **can** get work done on a machine you never heard of." *


"A distributed system is one where you **can** get work done on a machine you never heard of." *
(even when it crashes)


Start with People


People solve Problems

Thank you