

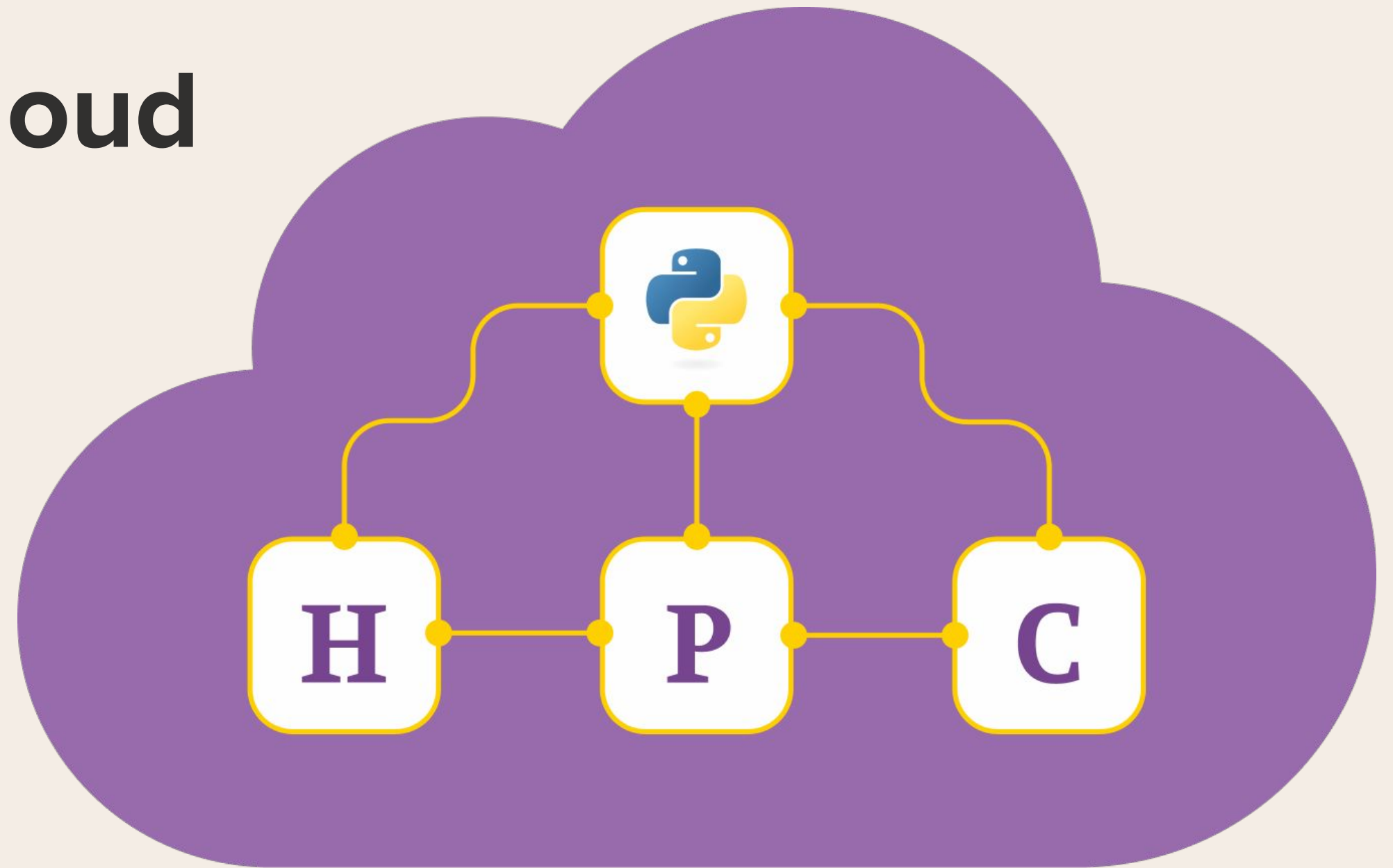
High Performance Computing in the Cloud

PyData Global

Dec 2023

Eddie Mattia

Outerbounds



Who this tutorial is for?

- I want to **make HPC more accessible** to many Pythonista data scientists
- I want to make HPC stack with **good enough performance** that is **easy to maintain** for my organization
- I want to learn about running **batch jobs on managed cloud services and/or Kubernetes**

Who this tutorial is not for?

- I want to optimize an HPC cluster from the hardware and kernels up
- I want to build HPC infrastructure at the scale of OpenAI

Part 1: HPC 🤝 AI 🤝 Cloud

Why is this a hot topic?

Part 2: What infrastructure makes my AI workflows go brrrr? 🧠

Comparing managed cloud solutions (e.g. AWS Batch) and cloud-agnostic solutions (e.g. Kubernetes)

Demo: Batch jobs with Kubernetes

Part 3: How to streamline ⚙️ and accelerate 🚀 the AI research experience?

Demo: Running Batch jobs from Python with Metaflow



Part 1: HPC 🤝 AI 🤝 Cloud

This section: **Motivating argument for HPC in the cloud**

1. Enterprises develop in the cloud. Clouds are becoming on-demand supercomputers.
2. Transformers improve with scale. Use OpenAI and develop muscle to build genAI without them.
 - Modern AI companies require similar infra primitives (batch jobs) as traditional HPC.

Remaining sections: **Setting up and enabling access to HPC primitives in the cloud**

1. Navigating a multi-cloud decision space.
2. Accessibility (from Python) for AI research teams.
 - Modern AI companies need to pick an abstraction layer for genAI devs, Python isn't a bad bet.

Enterprise teams develop in the cloud

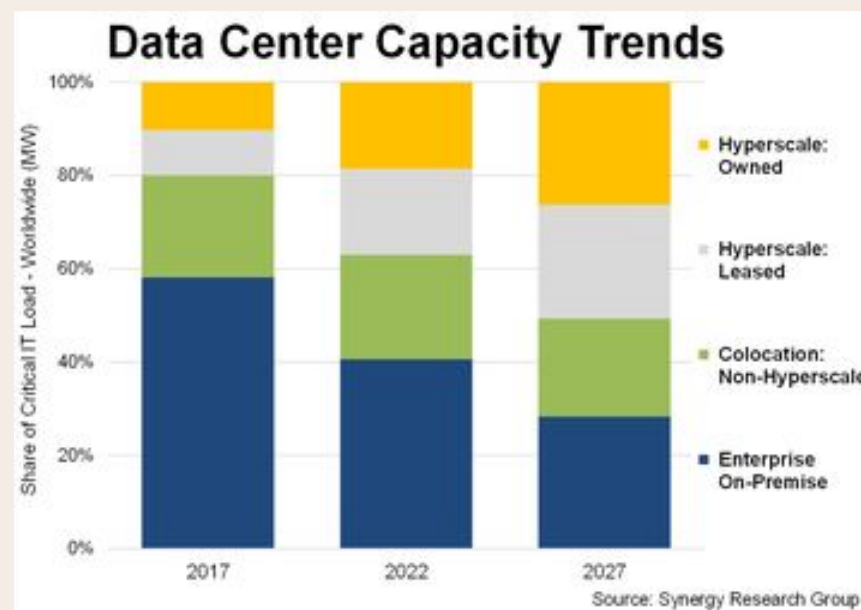
“Data center facilities consumes 3% of the global energy consumption and is soon expected to reach 8% mark.”

<https://www.bmc.com/blogs/cloud-growth-trends/>

Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly \$600 Billion in 2023

Cloud Drives Digital Business Transformation Through Emerging Technologies, Including Generative AI

<https://www.gartner.com/en/newsroom/press-releases/2023-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023>



<https://www.datacenterdynamics.com/en/opinions/on-prem-data-centers-arent-dead/>

The Pandemic Is Laying Waste to On-prem Data Centers

Major survey suggests half will be closed in the next two years.

<https://www.datacenterknowledge.com/cloud/pandemic-laying-waste-prem-data-centers>

**Development teams need to build AI systems in the 2020s
... and need the knowledge and muscle to train their own models!**

The economic potential of generative AI: The next productivity frontier

<https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier>

AI model training rekindles interest in on-premises infrastructure

<https://siliconangle.com/2023/10/16/ai-model-training-rekindles-interest-premises-infrastructure/>

Your Personal Information Is Probably Being Used to Train Generative AI Models

<https://www.scientificamerican.com/article/your-personal-information-is-probably-being-used-to-train-generative-ai-models/>

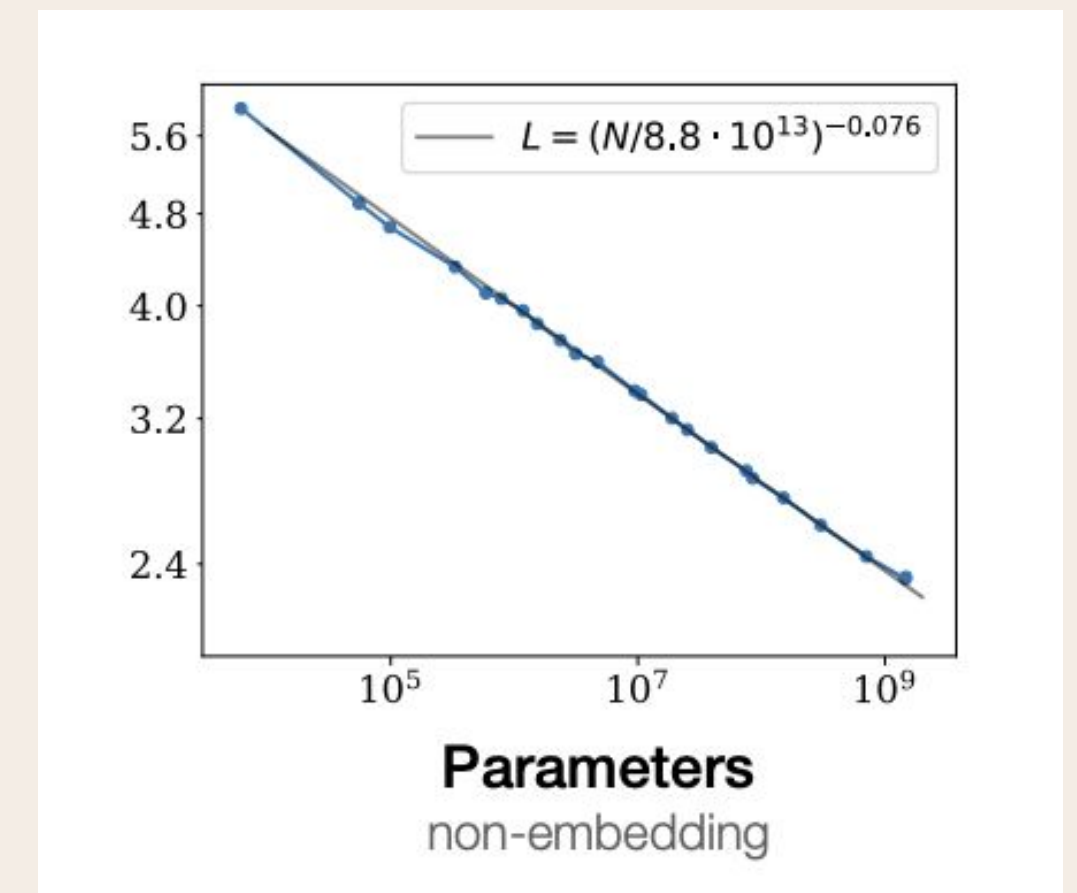
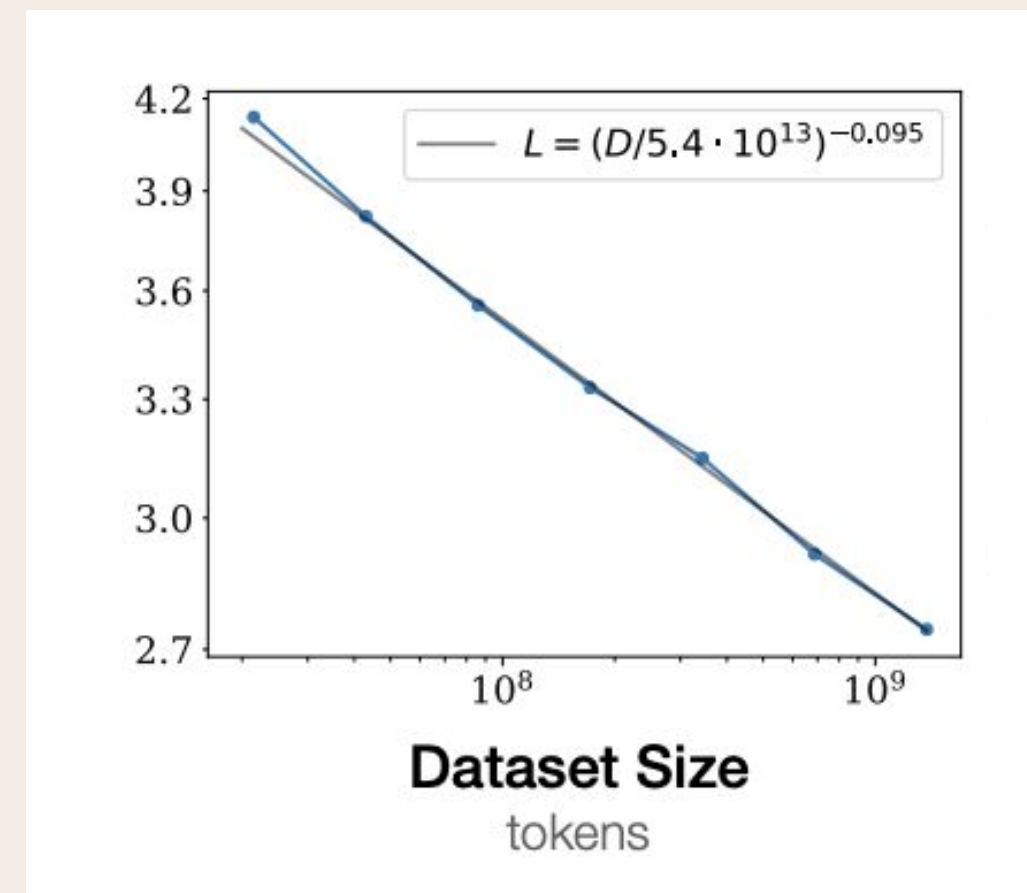
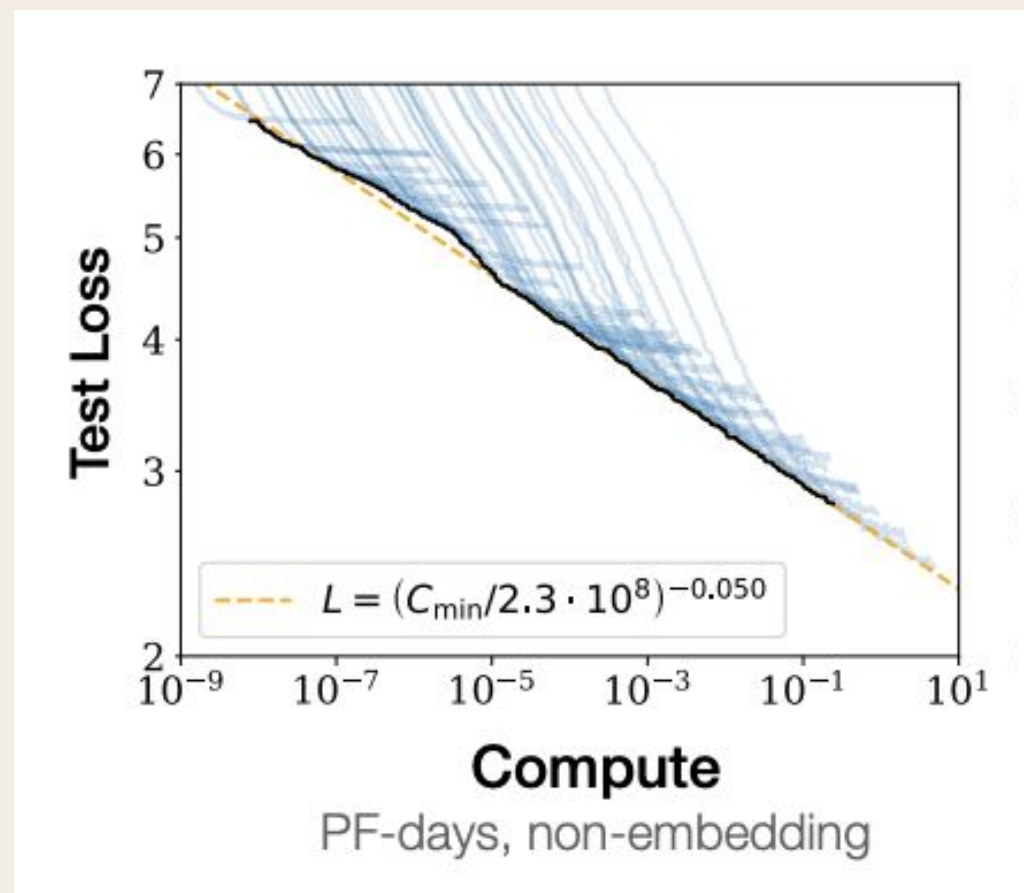
The copyright issues around generative AI aren't going away anytime soon

<https://techcrunch.com/2023/09/21/the-copyright-issues-around-generative-ai-arent-going-away-anytime-soon/>

“Moving 100 petabytes of data to the cloud is next to impossible,” said Anand Babu Periasamy, co-founder and CEO of MinIO Inc. “They’re going to move AI to where data is.”

Ways to make deep learning models do better (holding algorithms constant)

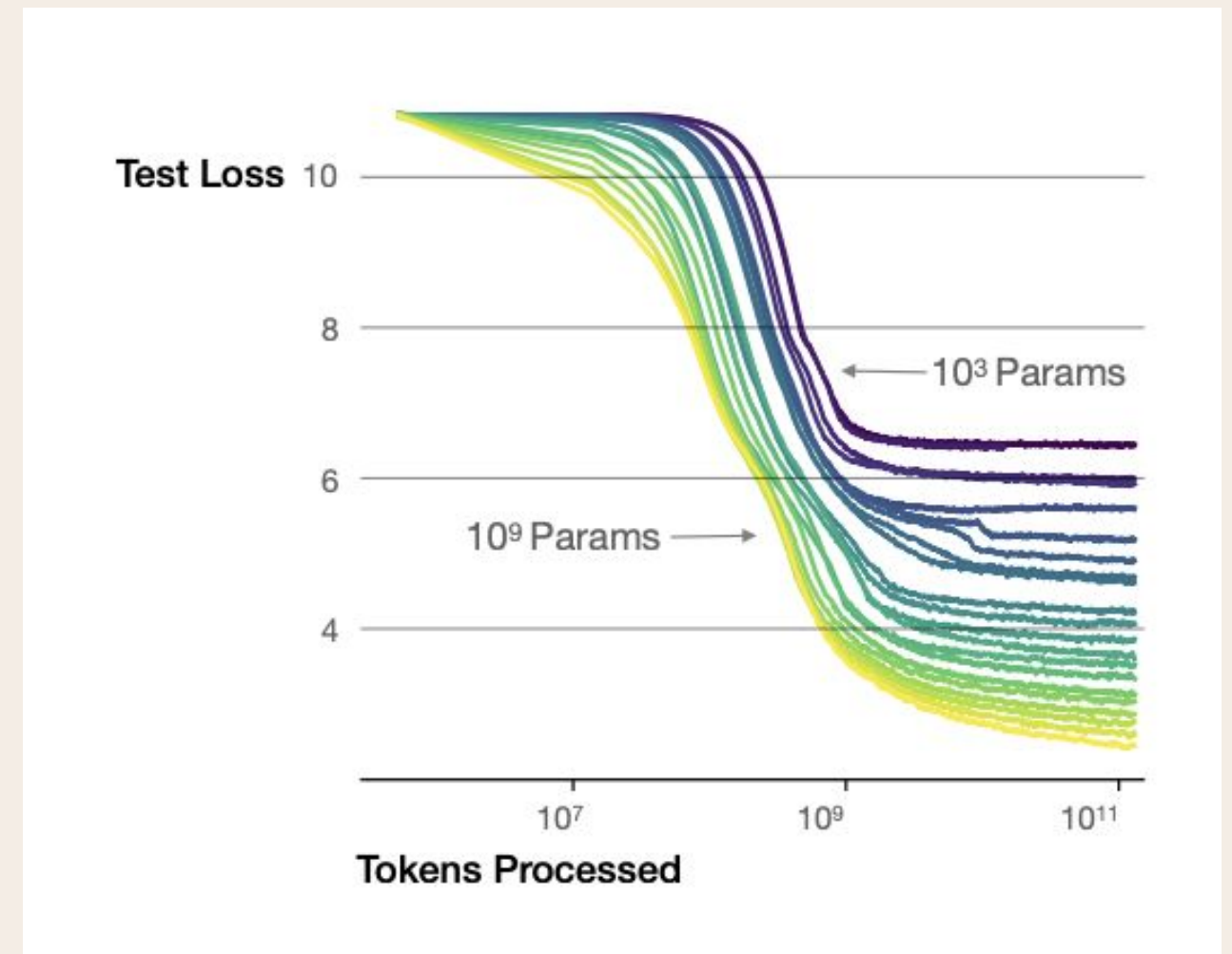
- More N model params
- More D tokens in training dataset
- More C compute (FLOPs)



Ways to make deep learning models do better (holding algorithms constant)

- More N model params
- More D tokens in training dataset
- More C compute (FLOPs)

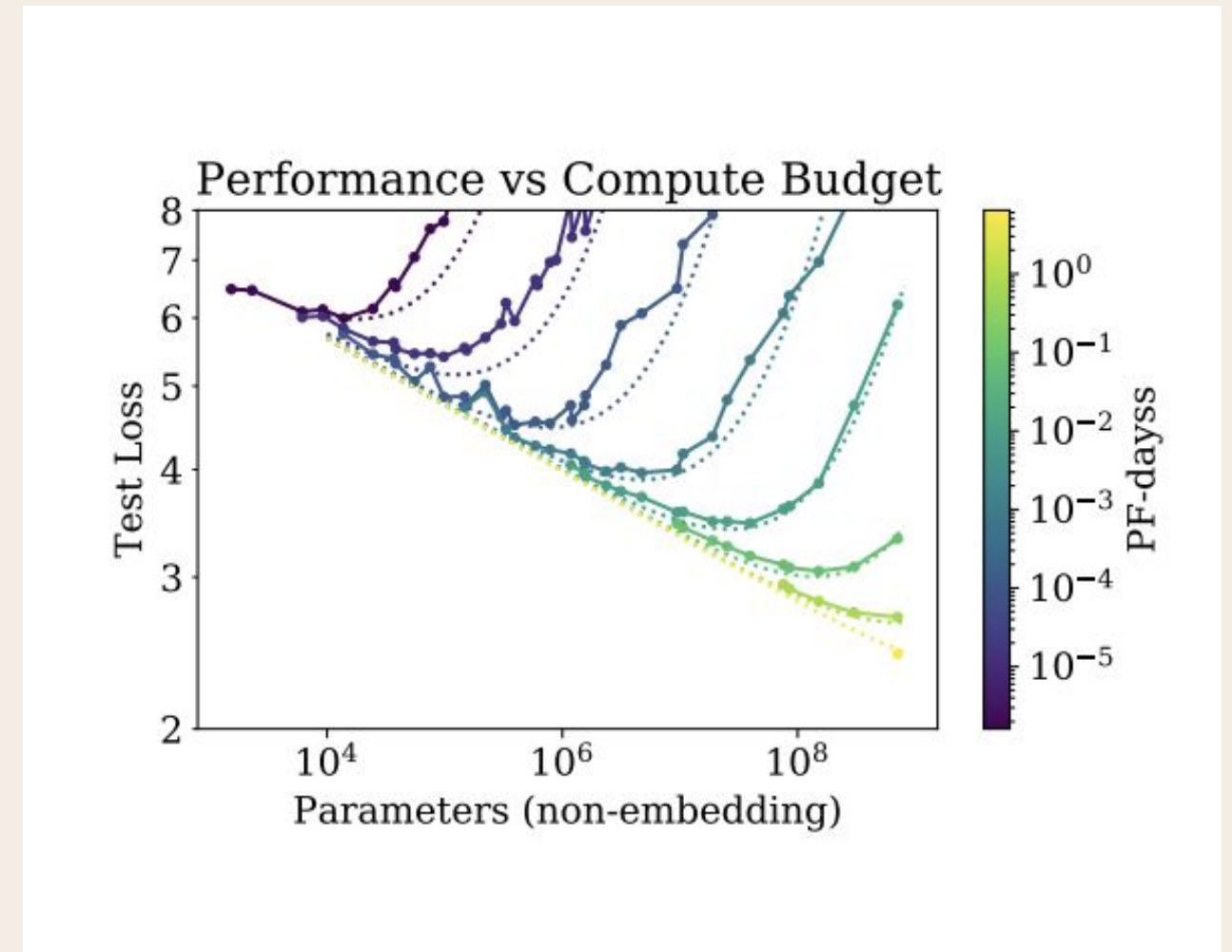
Sample efficiency: as N grows, same performance achieved after model learns from fewer samples in D



Ways to make deep learning models do better (holding algorithms constant)

- More N model params
- More D tokens in training dataset
- More C compute (FLOPs)

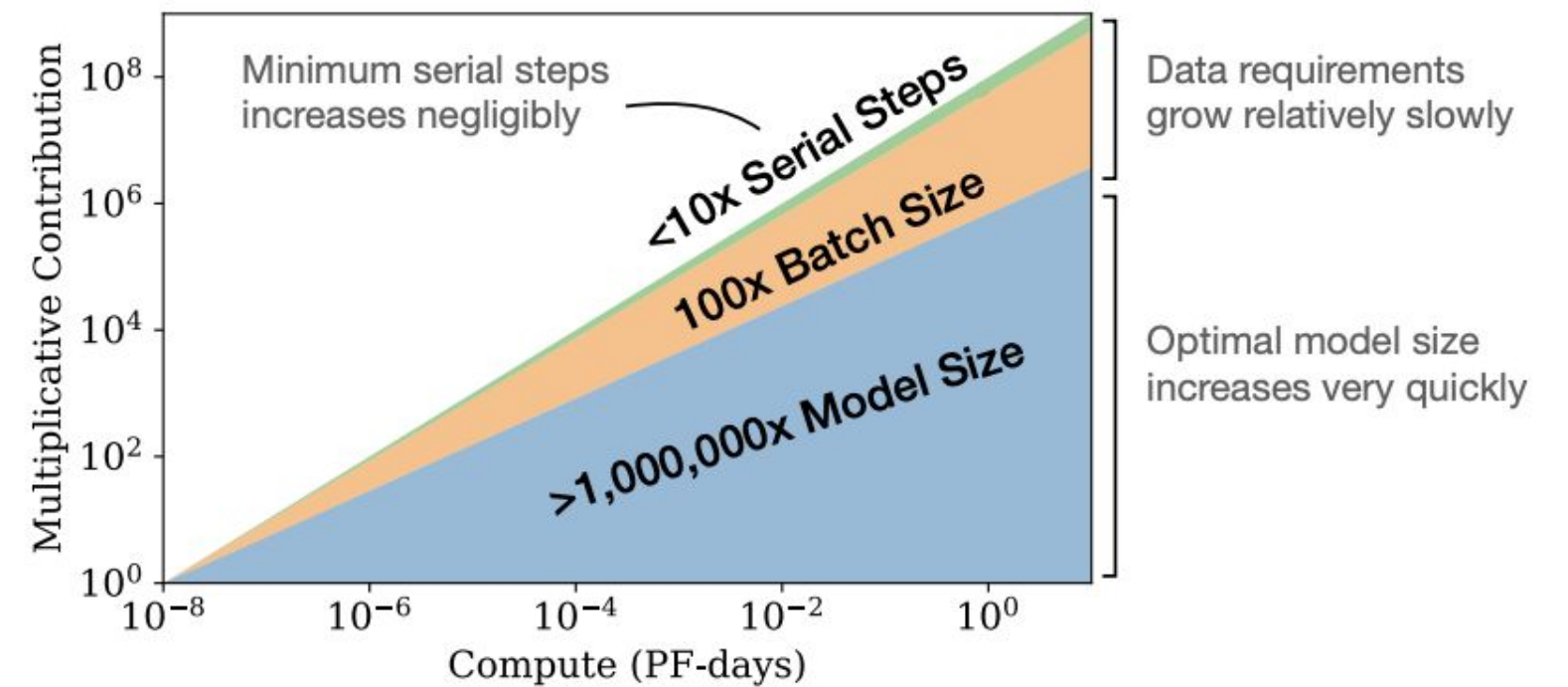
Optimal N grows with C



Ways to make deep learning models do better (holding algorithms constant)

- More **N** model params
- More **D** tokens in training dataset
- More **C** compute (FLOPs)

Claim: Optimal **N** grows with **C** faster than optimal growth in **D**

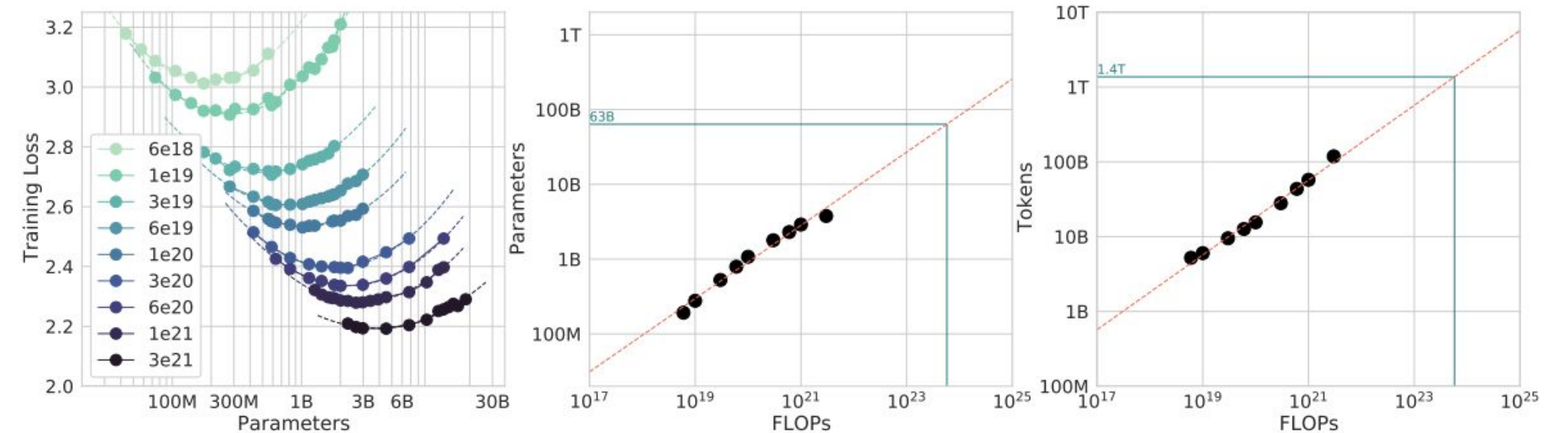


Ways to make deep learning models do better (holding algorithms constant)

- More N model params
- More D tokens in training dataset
- More C compute (FLOPs)

Chinchilla paper

- increasing D is equally as important as increasing N

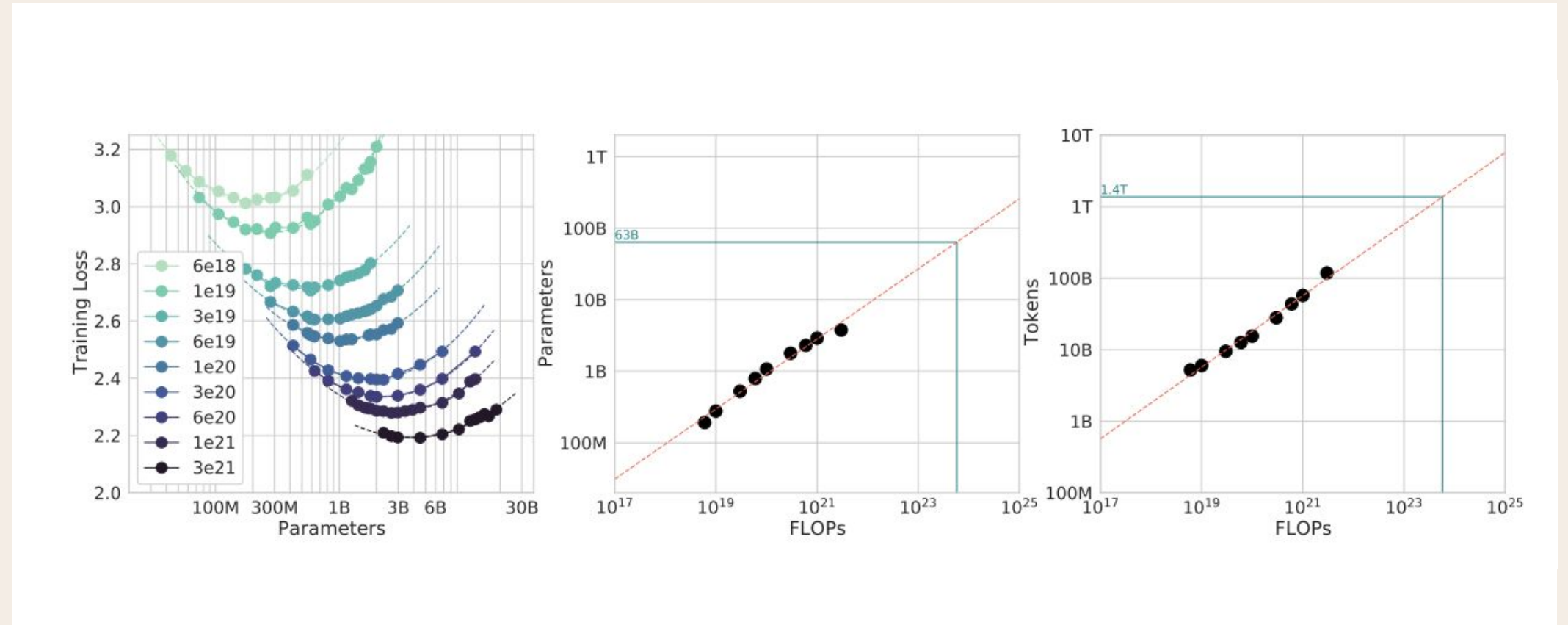


Ways to make deep learning models do better (holding algorithms constant)

- More N model params
- More D tokens in training dataset
- More C compute (FLOPs)

Chinchilla paper

- increasing D is equally as important as increasing N
- for each C , there is a clearly optimal N

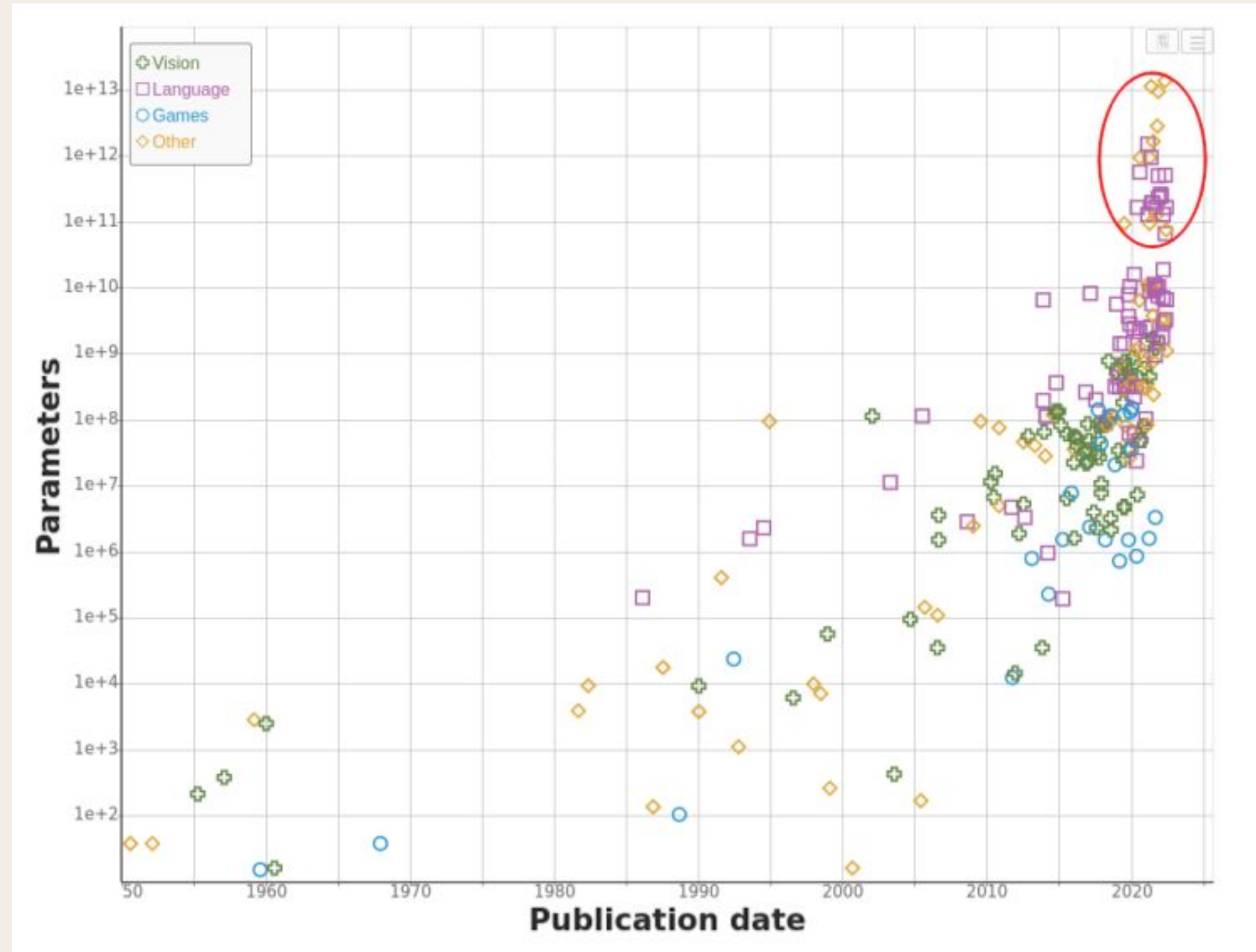


Ways to make deep learning models do better (holding algorithms constant)

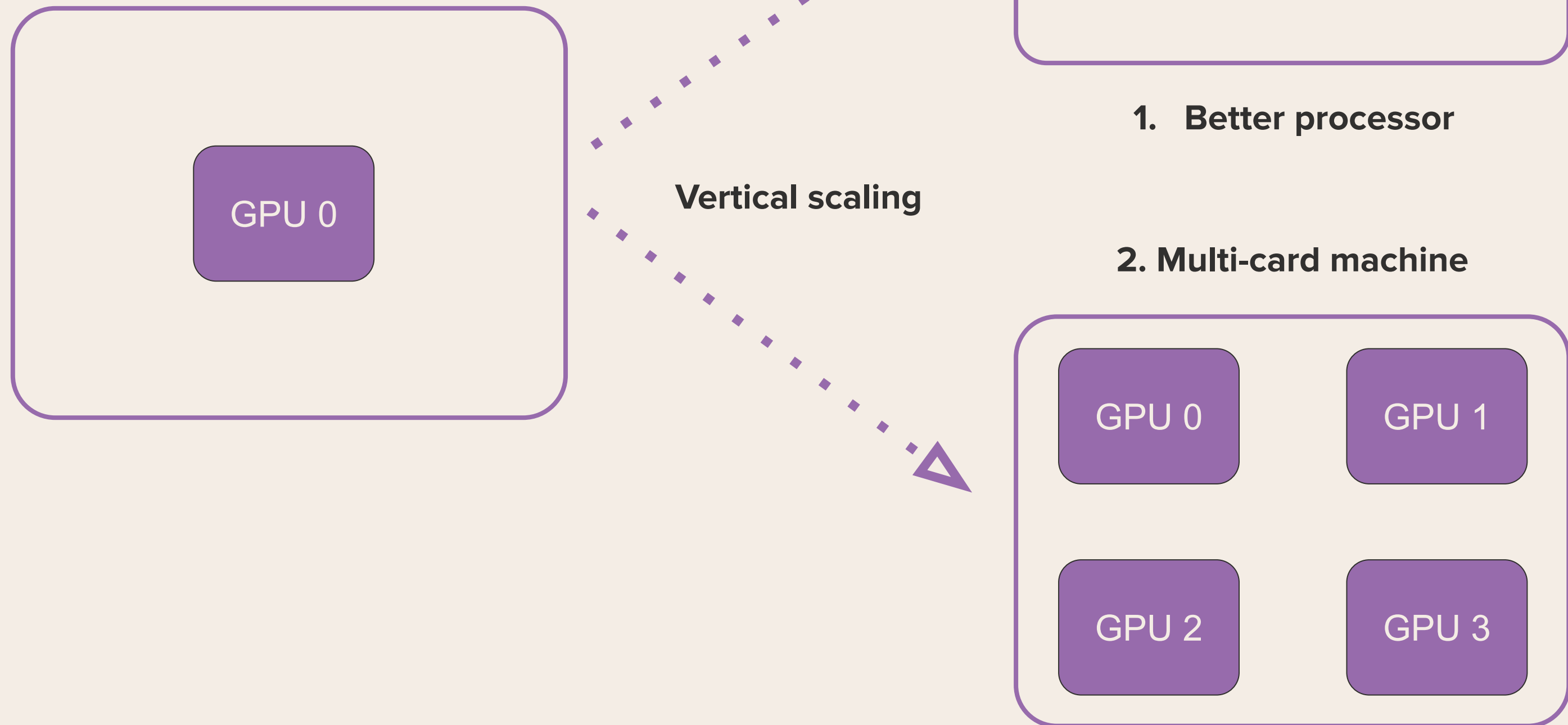
- More N model params
- More D tokens in training dataset
- More C compute (FLOPs)

Chinchilla paper

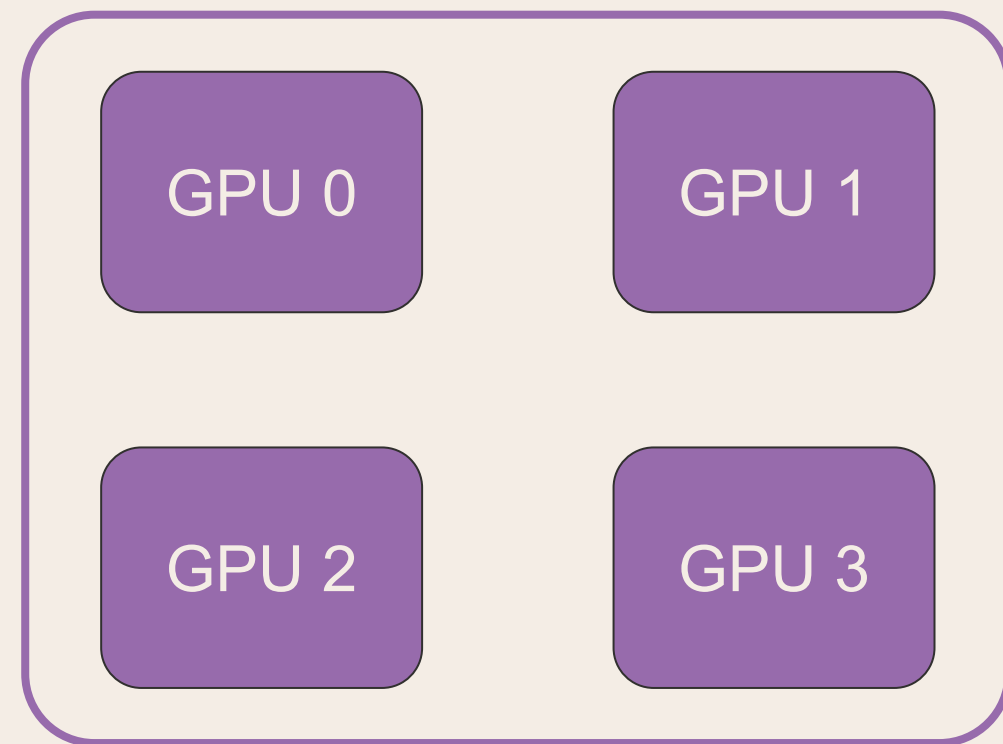
- increasing D is equally as important as increasing N
 - for each C , there is a clearly optimal N
- more sample-efficient learning requires more C
- if you cannot readily increase C ,
eventually expect worse performance as N grows



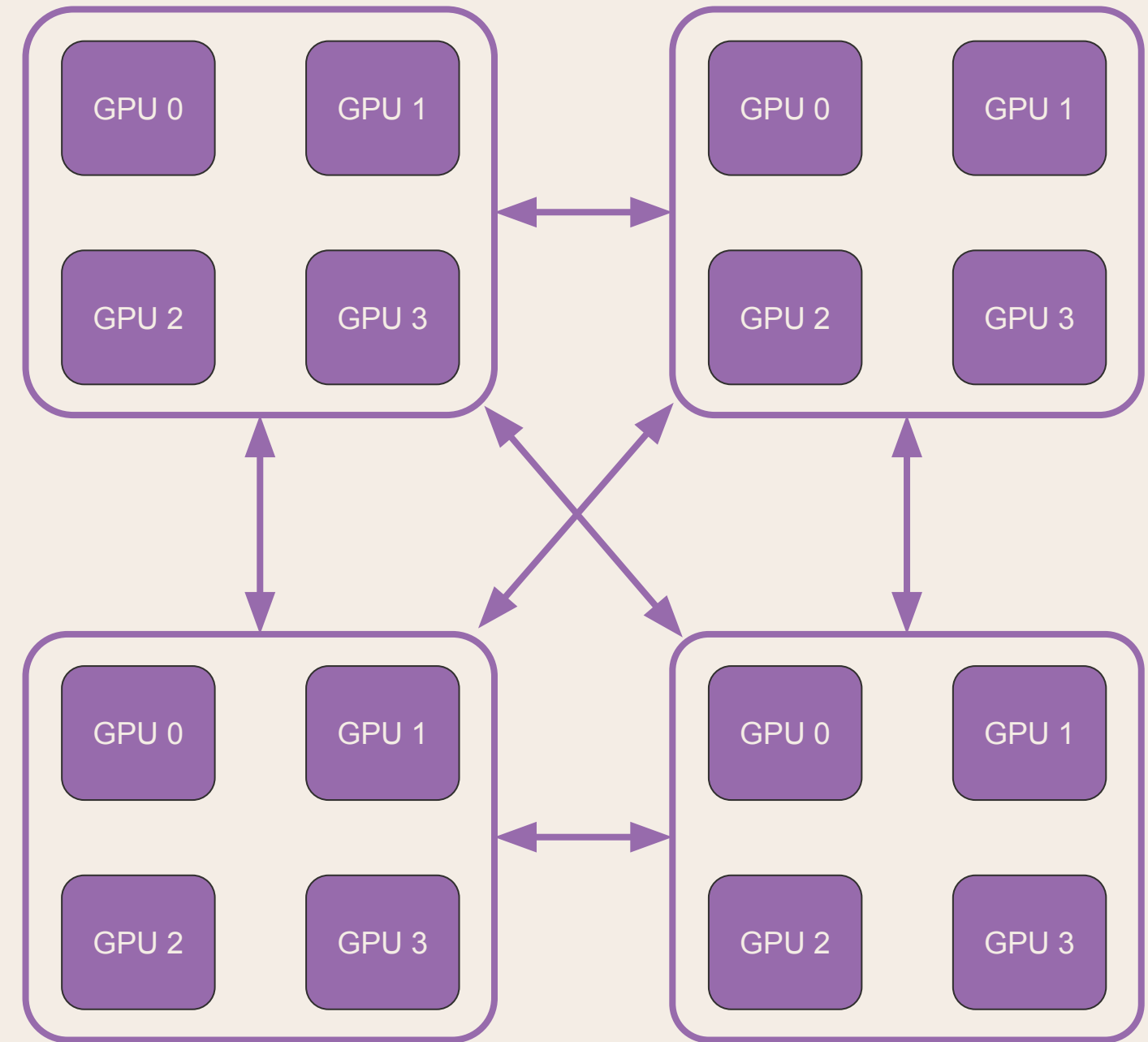
How to increase C?



How to increase C?

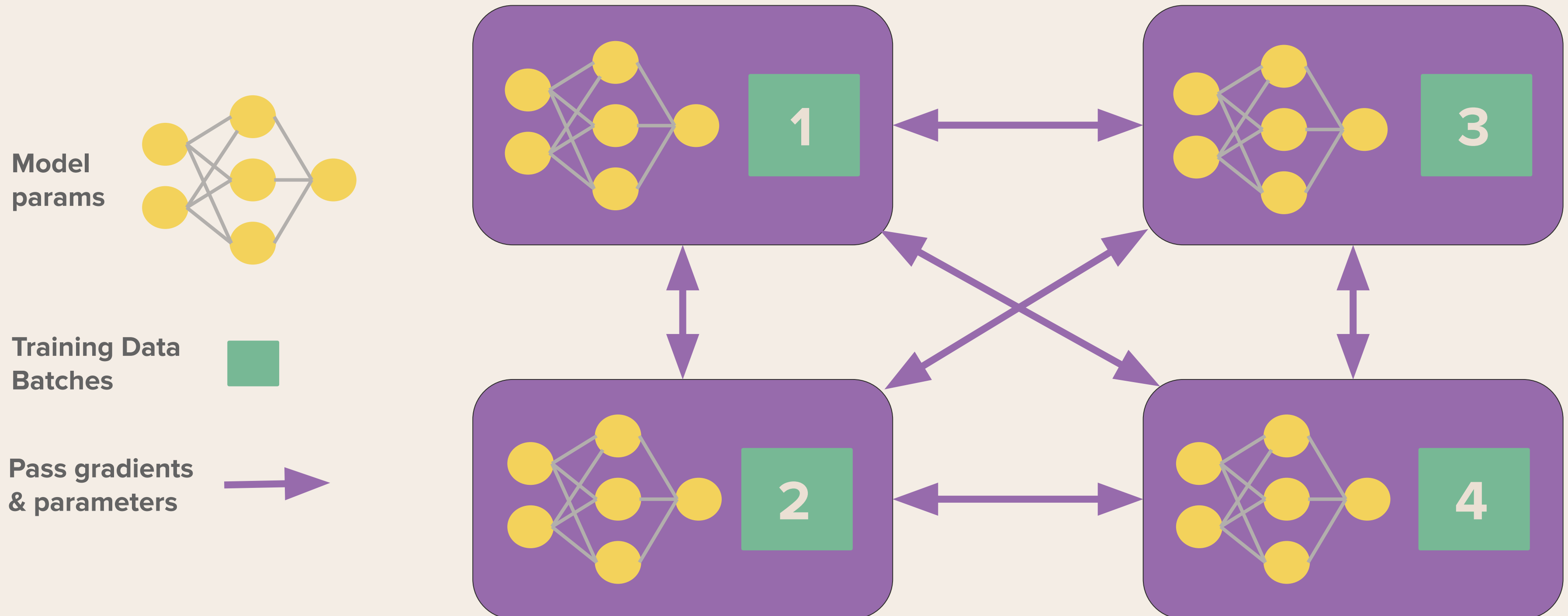


**Horizontal /
HPC scaling**



Data Parallelism

When D is too big for VRAM or you have extra C



Data Parallelism

When D is too big for VRAM or you have extra C

```
import torch.distributed as dist
import torch.multiprocessing as mp
from torch.nn.parallel import DistributedDataParallel as DDP

def train(RANK, WORLD_SIZE):
    dist.init_process_group("gloo", rank=RANK, world_size=WORLD_SIZE)
    ...
    model = NeuralNet().to(RANK) # put WHOLE MODEL on this device
    ddp_model = DDP(model, device_ids=[RANK])
    ...
    dist.destroy_process_group()

if __name__ == '__main__':
    train(MY_RANK, WORLD_SIZE)
```

Data Parallelism

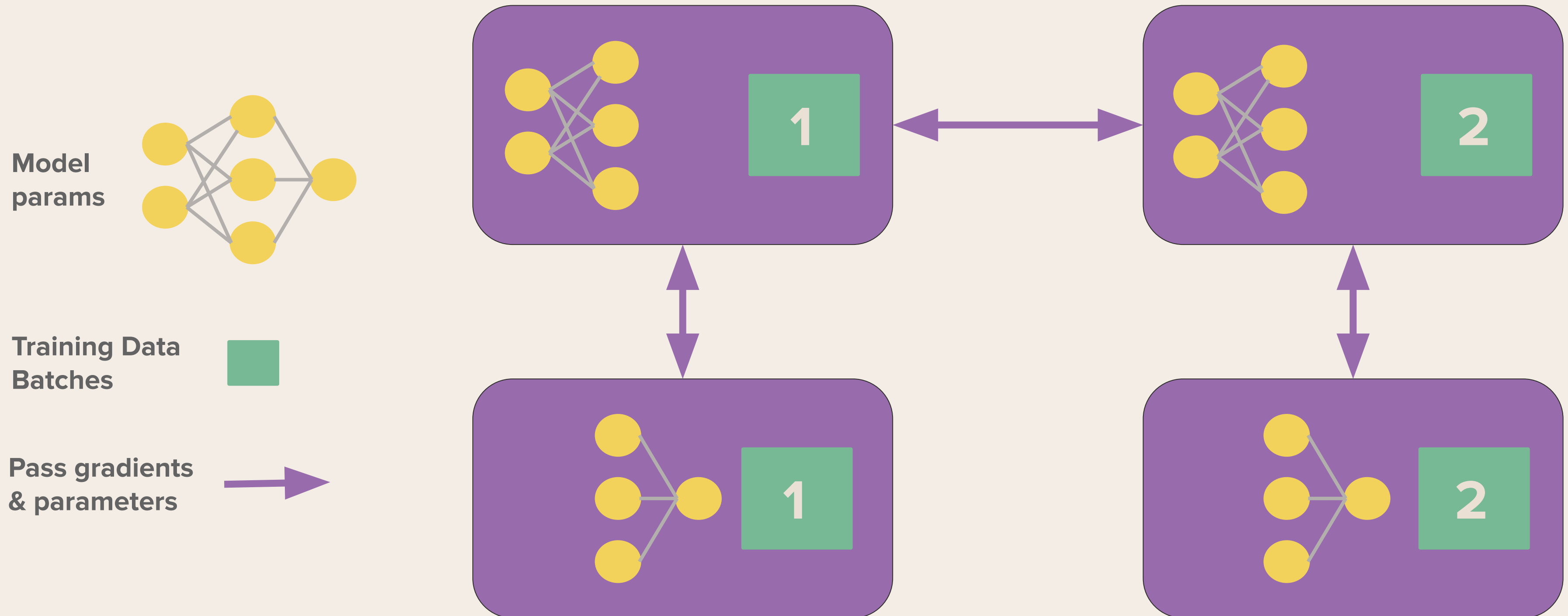
When D is too big for VRAM or you have extra C

Concepts

- shard data → **independent** forward pass → gradient sync → **independent** backpropagation
- effective batch size = batch_size * world_size (num GPU workers)
- hard to use this in Notebook, because DDP implementations call Python processes under the hood

Model Parallelism

When N is too big for VRAM



Model Parallelism

When N is too big for VRAM

```
self.seq0 = nn.Sequential(  
    self.conv1, self.bn1, self.relu, self.maxpool, self.layer1, self.layer2)  
    .to('cuda:0') # sends the first sequence of the model to the first GPU  
  
self.seq1 = nn.Sequential(self.layer3, self.layer4, self.avgpool)  
    .to('cuda:1') # sends the next sequence of the model to the second GPU
```

Model Parallelism

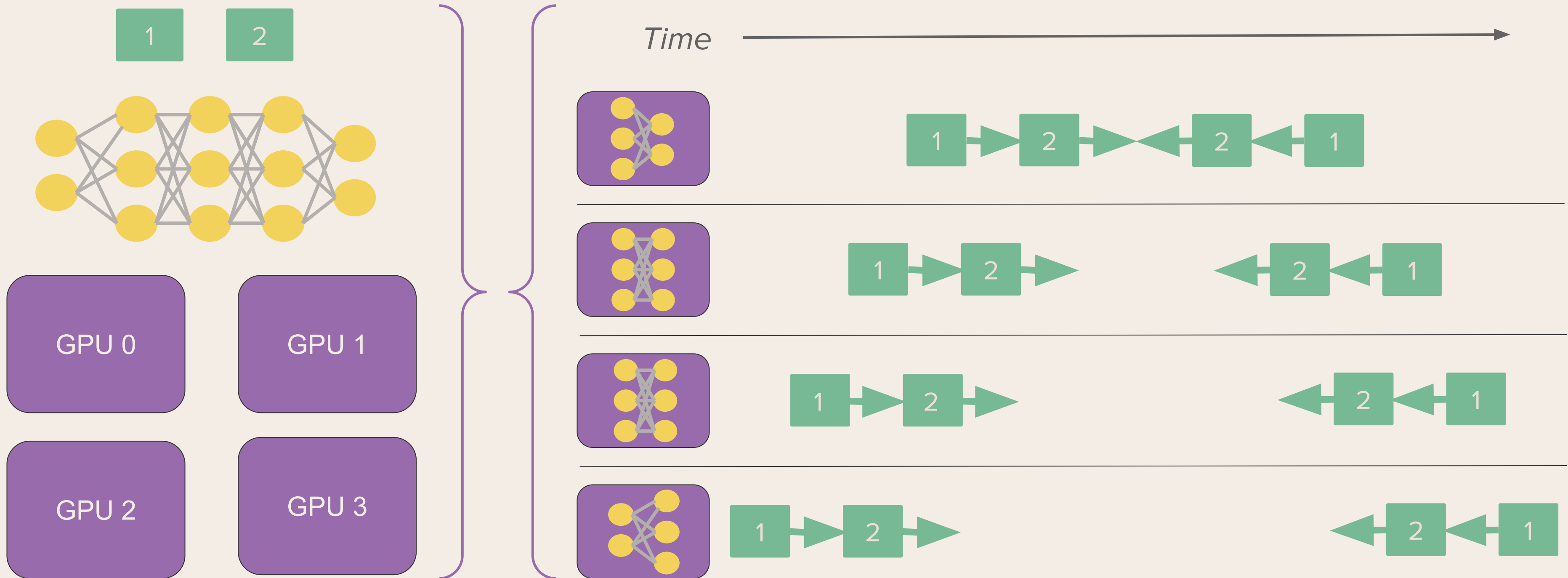
When N is too big for VRAM

Concepts

- **dependent** forward pass → gradient sync → **dependent** backpropagation
- effective batch size = batch_size * world_size (num model parallel GPU worker groups)
- “scope” model layers to different devices
- [Combining DDP with Model Parallelism](#)

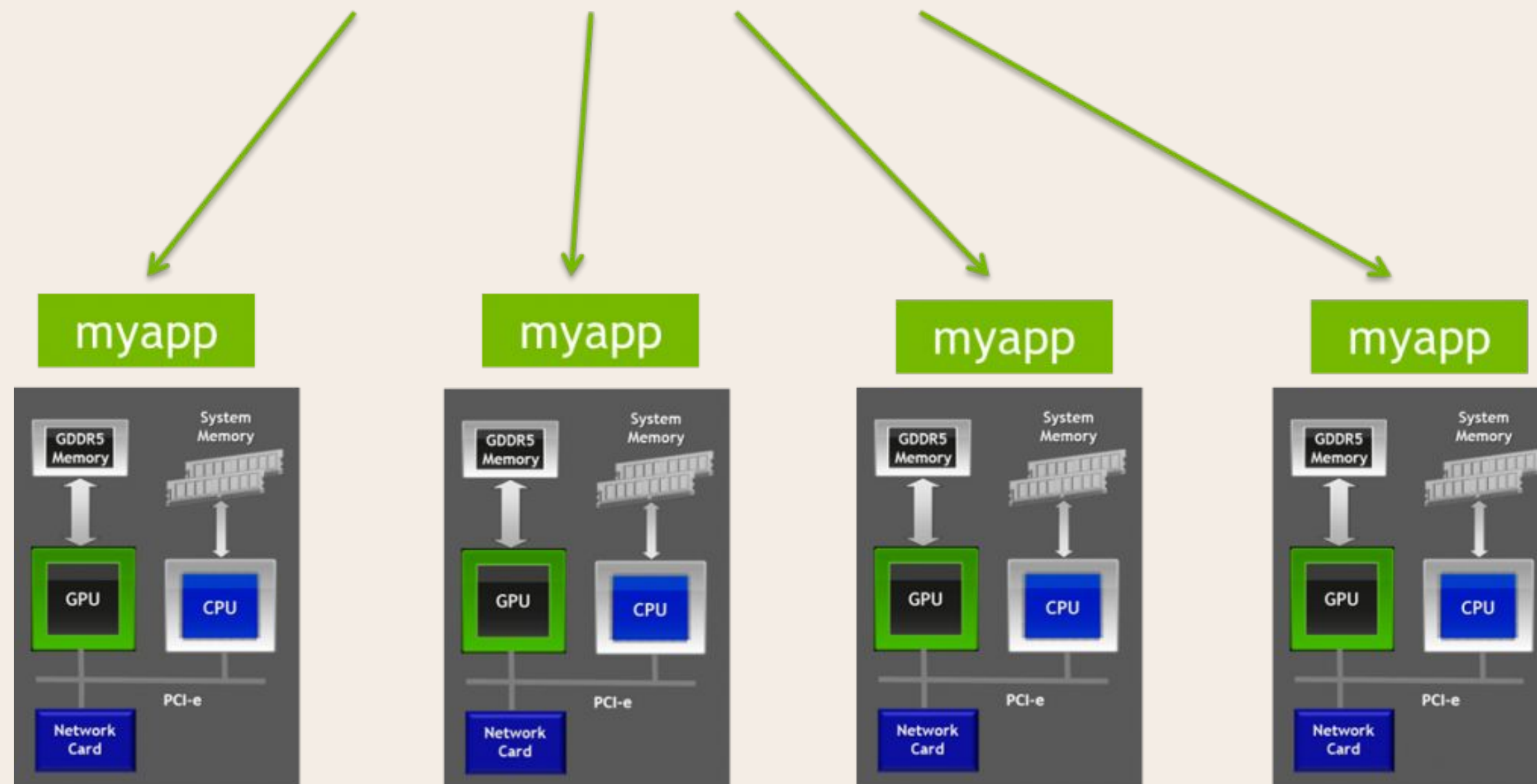
Pipeline Parallelism

To make model parallelism go brrrr 🌀



**MPI has existed to do jobs like this since before the internet.
Modern AI frameworks still use it.**

```
mpirun -np 4 ./myapp <args>
```



How to run a distributed training job on N_NODES? Many ways to run the same PyTorch code!

torchrun

```
--nnodes=1:$N_NODES  
--nproc-per-node=$NUM_TRAINERS  
--backend=ncc1  
...  
train.py --arg1 X
```

- run on every node
- Torch distributed takes care of comms, so long as [torch backend](#) is configured
 - GLOO for CPU, NCCL for GPU
- run any torch.distributed program

[\[torchrun Documentation\]](#)

mpirun

```
-np=$N_NODES_BY_NUM_TRAINERS  
...  
python train.py --arg1 X
```

- run on main node only
- requires passwordless SSH between nodes
- nodes know each other through a “host file”
- run any executable, not just Python

[\[mpirun Documentation for OpenMPI\]](#)

deepspeed

```
--num_nodes=$NUM_NODES  
...  
train.py --arg1 X  
...  
--deepspeed_config ds_config.json
```

- run on main node only
- uses OpenMPI → hostfile pattern
- NCCL backend for communication
- run any torch.distributed program, modified to run with deepspeed

[\[deepspeed Documentation\]](#)

Separation of concerns

AI research gets blocked by researchers context switching between PyTorch model code and the compute cluster they run a torchrun/deepspeed/MPI command on.

AI researchers work in teams

Managing separate compute for each scientist introduces OpEx for platform engineers.

Scientists don't want another job to block their work getting done.



Part 2: AI Training Infrastructure in the Cloud

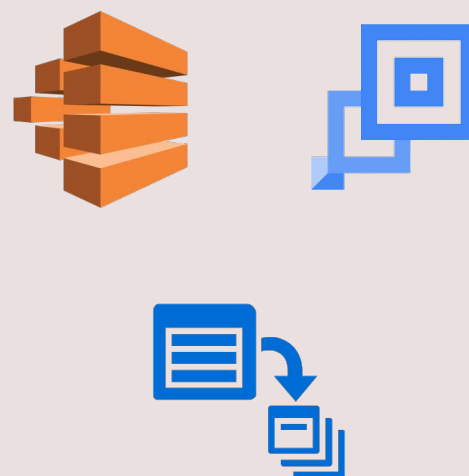
effort

*Effects of a good
AI development
platform*

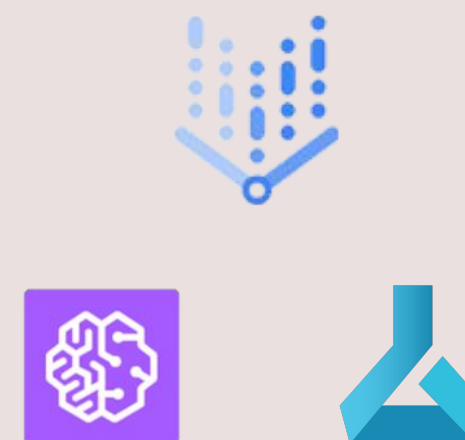
IaaS & DIY
infrastructure



hyperscaler PaaS



hyperscaler SaaS



Batch compute
options in a
hybrid-cloud world

High flexibility

Medium
flexibility

Low flexibility

OpEx
cost

JTBD / Solution	Slurm	Kubernetes	Hyperscaler PaaS (e.g., AWS Batch)	Hyperscaler SaaS (e.g., Sagemaker)
<i>How to form a cluster?</i>	Install munge, install slurm. run slurmd on control, slurmd on compute nodes.	Install kubectl & kubeadm init on control. kubeadm join on others. Run kubelet.	Automated with multi-node job submission	Automated
<i>How to maintain job queue?</i>	Define partitions (node groups) in slurm.conf.	Volcano. Yunikorn. Armada. Many CRDs you can try for this.	Attach policy on creating a job queue & connecting to compute environment	Automated
<i>How to write research code?</i>	Script submitted to e.g., mpirun command	Package in container run command	Package in container run command	Python SDK
<i>How to submit a job?</i>	Run sbatch command with a Slurm shell script	Run kubectl command with a job configuration in a yaml file.	Use Python SDK (boto3) or AWS cli	Python SDK

JTBD / Solution	Slurm	Kubernetes	Hyperscaler PaaS (e.g., AWS Batch)	Hyperscaler SaaS (e.g., Sagemaker)
<i>How to access data?</i>	DIY and/or ask sysadmin to mount filesystem (e.g., NFS)	DIY with secrets and/or ask sysadmin to mount files (e.g.,persistent volumes)	Configure IAM to read from S3 / RDS / etc.	Configure IAM to read from S3 / RDS / etc.
<i>How to add compute to resources pool?</i>	Run slurmd daemon on compute node	Register a Kubernetes node with control plane	EC2 instances in AWS Batch compute environment	Special 🪂 EC2 instances
<i>Container-based?</i>	Yes, but typically not docker. Singularity is most common.	Yes	Yes	Yes
<i>Interprocess communication?</i>	PMI2 or framework-specific (e.g., SSH for MPI)	CNI plugins: e.g., Calico, Flannel, Weave. Framework-specific (e.g., SSH for MPI)	AWS security groups + framework-specific (e.g., SSH for MPI)	Automated

DIY path: Kubernetes vs Slurm

High-level differences

- Kubernetes focuses on dynamically scaling clusters and cloud-native features
- Slurm focuses on job scheduling for static clusters and HPC features

Slurm partitions and Kubernetes node groups are both groups of computers in the respective cluster.

- Organizing compute resources by intended use and characteristics (e.g., GPU nodes)

Slurm has sophisticated scheduling algorithms and priority settings for multi-user HPC environments.

- Examples: [Multifactor priority](#), [Fairshare priority](#), [Gang scheduling](#), [Fractional GPU](#)

Slurm is more established for HPC and AI training.

Kubernetes has more developer ecosystem momentum, and many more use cases beyond HPC.





Does Kubernetes scale for serious AI training?

We've scaled Kubernetes clusters to 7,500 nodes, producing a scalable infrastructure for large models like GPT-3, CLIP, and DALL·E, but also for rapid small-scale iterative research such as Scaling Laws for Neural Language Models.

<https://openai.com/research/scaling-kubernetes-to-7500-nodes>

“To create Grok, we built a custom training and inference stack based on Kubernetes, Rust and JAX.”

<https://x.ai/>

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	Aurora - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	4,742,808	585.34	1,059.33	24,687
3	Eagle - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84	
		New to top 10			 

Demo / case study : Kubernetes cluster as compute provider

“Forming a cluster” means many Kubernetes nodes pooling the underlying computers together.

- Let's do it on the laptop with minikube.

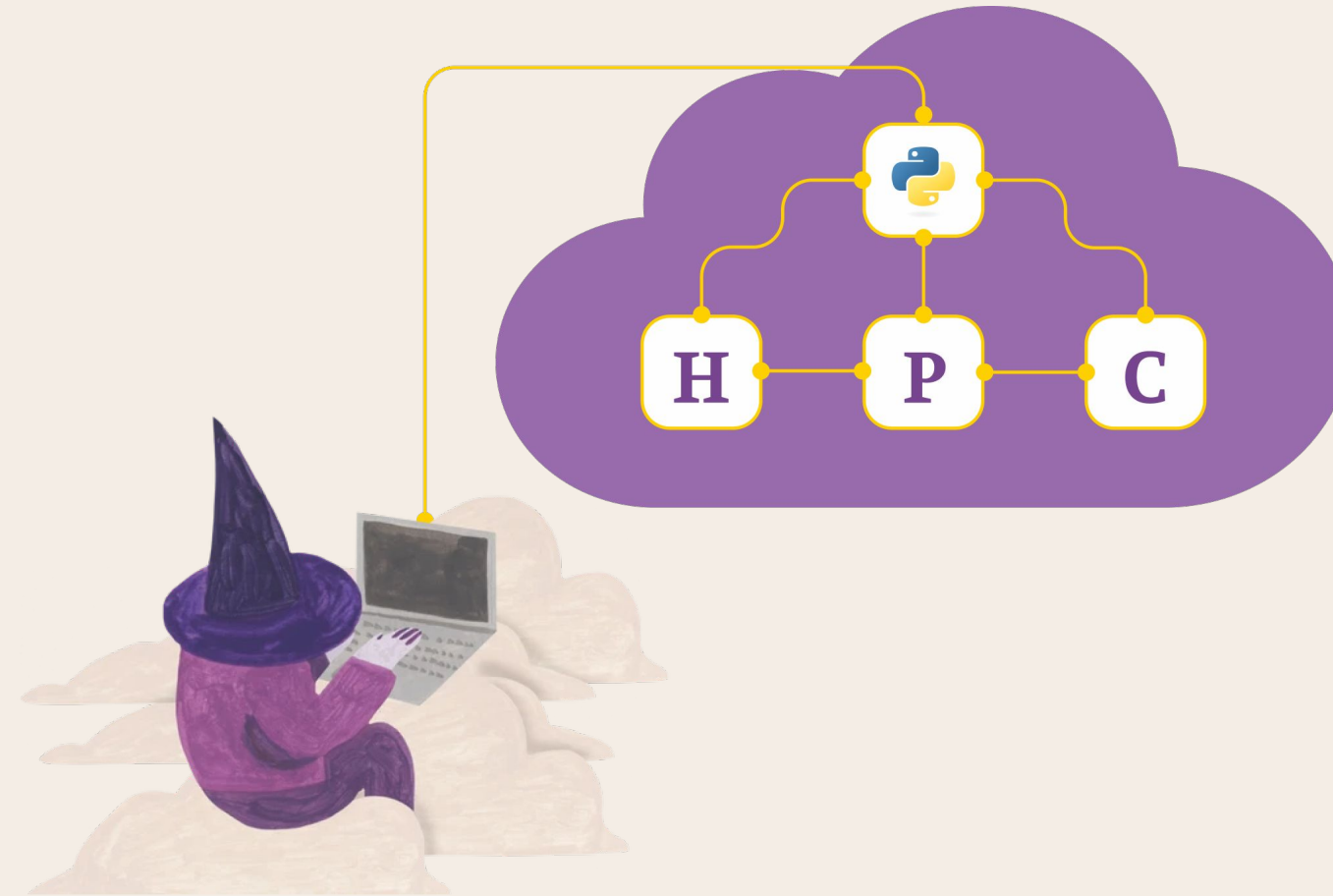
“Submitting a job” means running a Kubernetes job (or related variant) on the cluster's nodes.

- Let's do a simple example.

“Queuing a job” means knowing what hardware is in the cluster, and assigning the job appropriately.

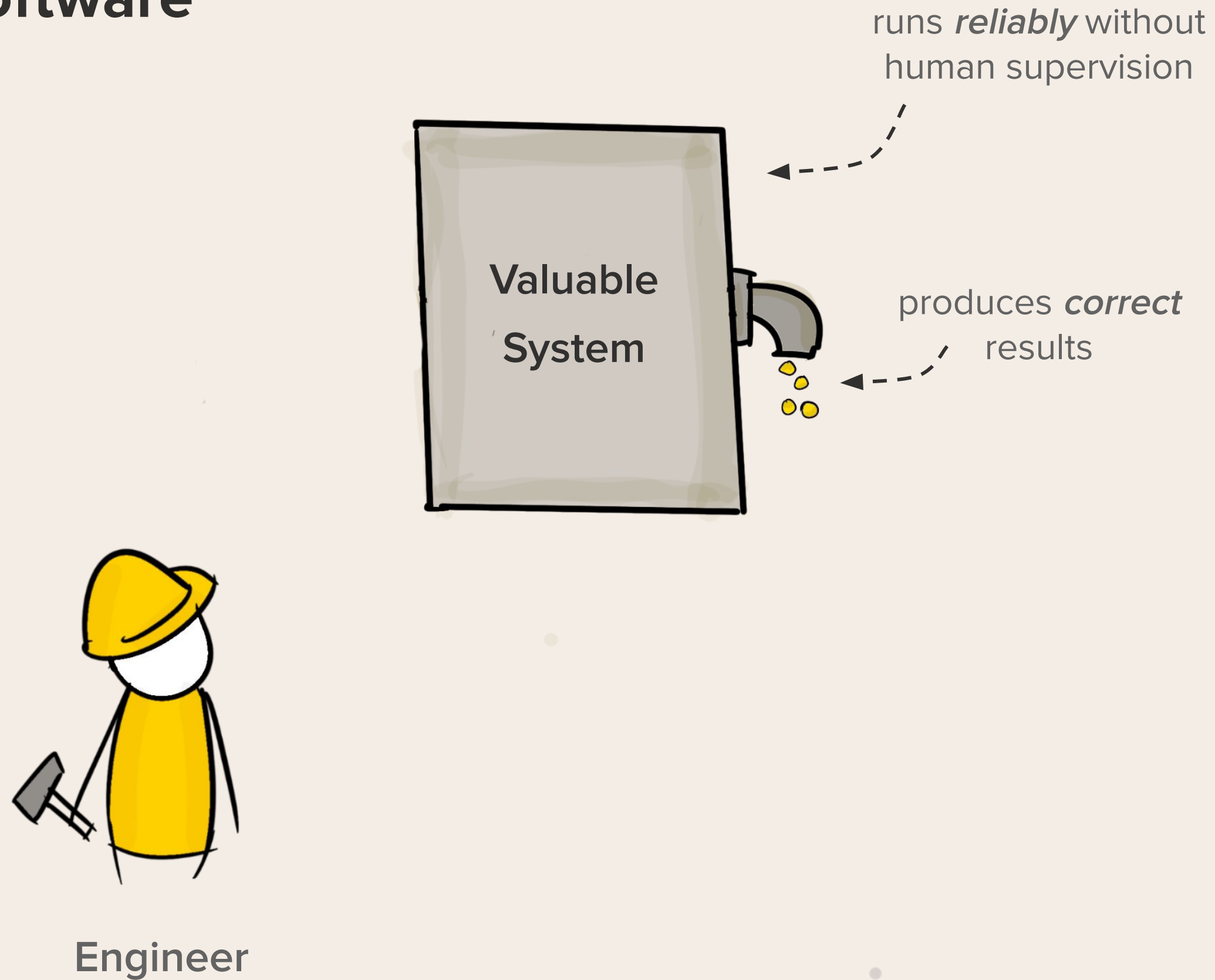
- Let's do it on the minikube cluster with Volcano.
- Let's submit a PyTorch example using torchx.

DEMO → <https://github.com/outerbounds/pydata-global-2023-hpc/tree/main/torchx-local>

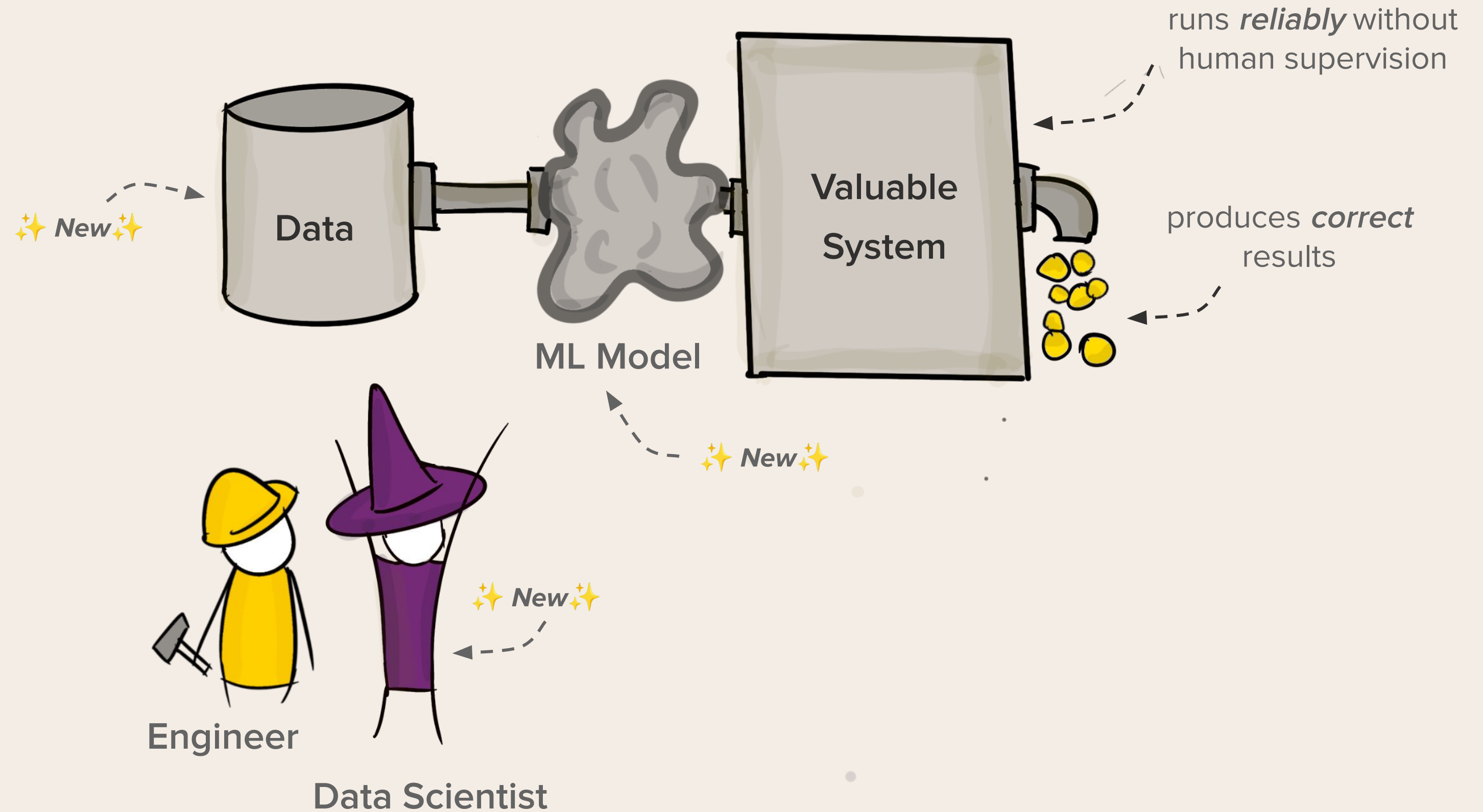


Streamlining the AI research experience

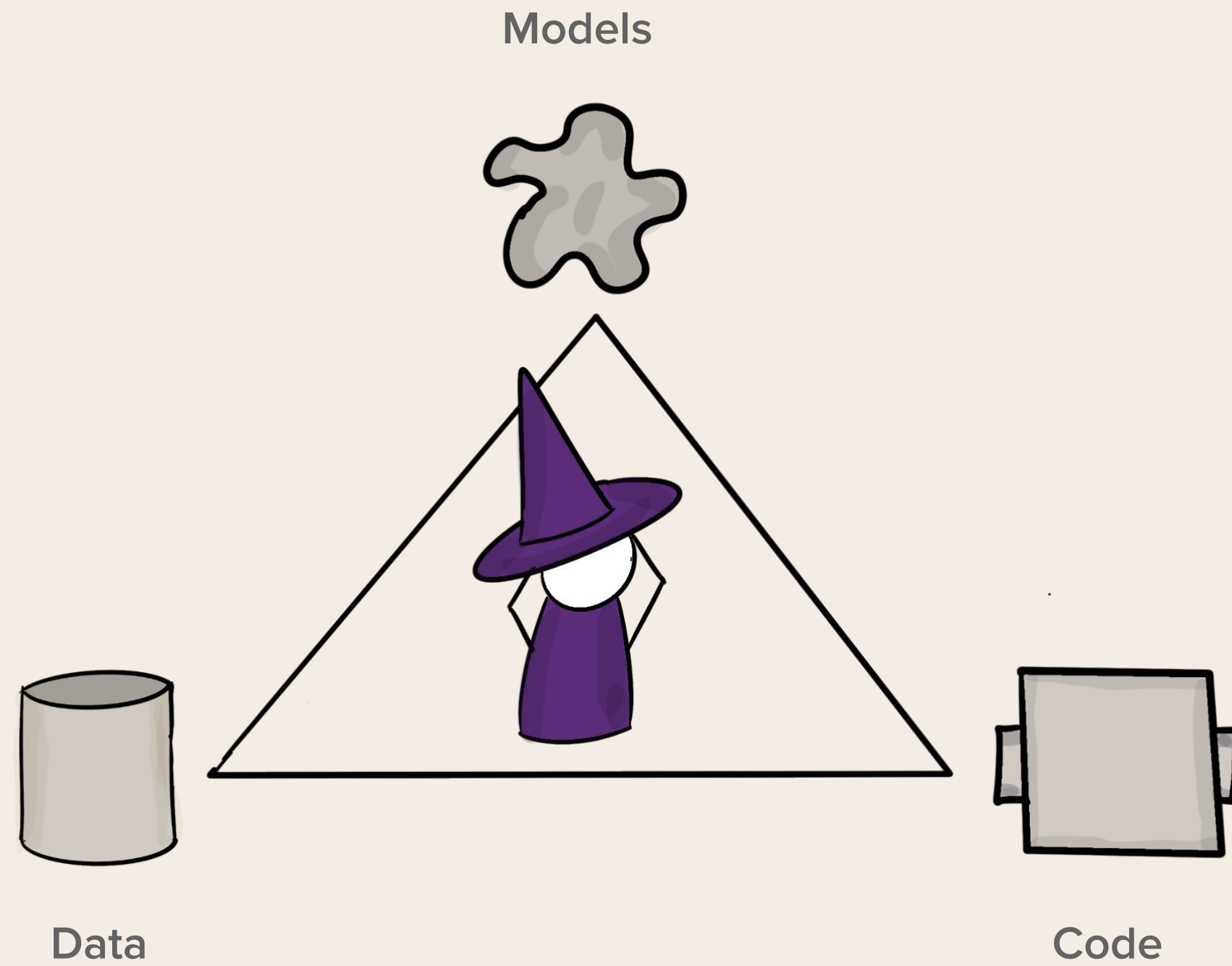
Traditional software



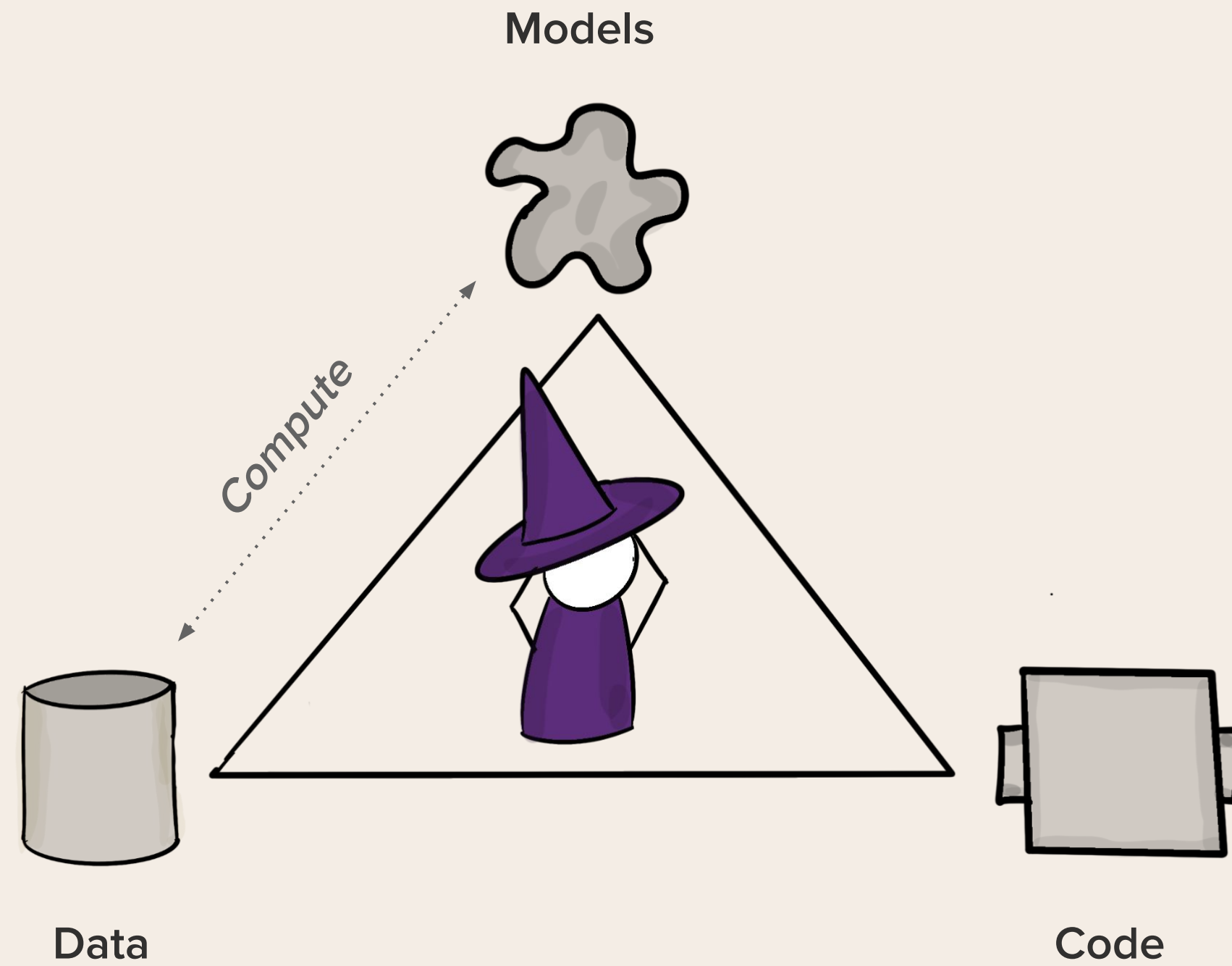
ML-powered software



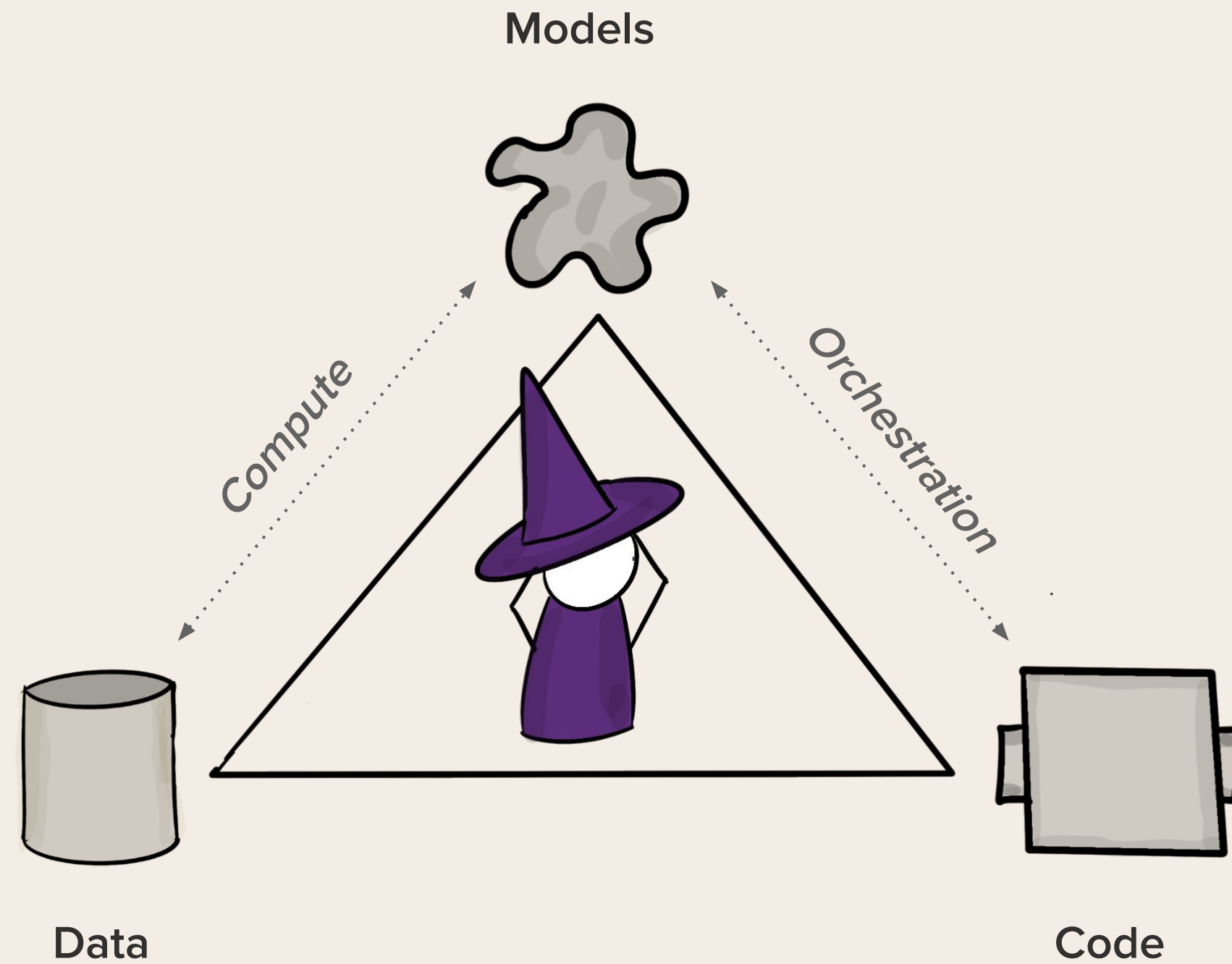
Building Blocks of ML Systems



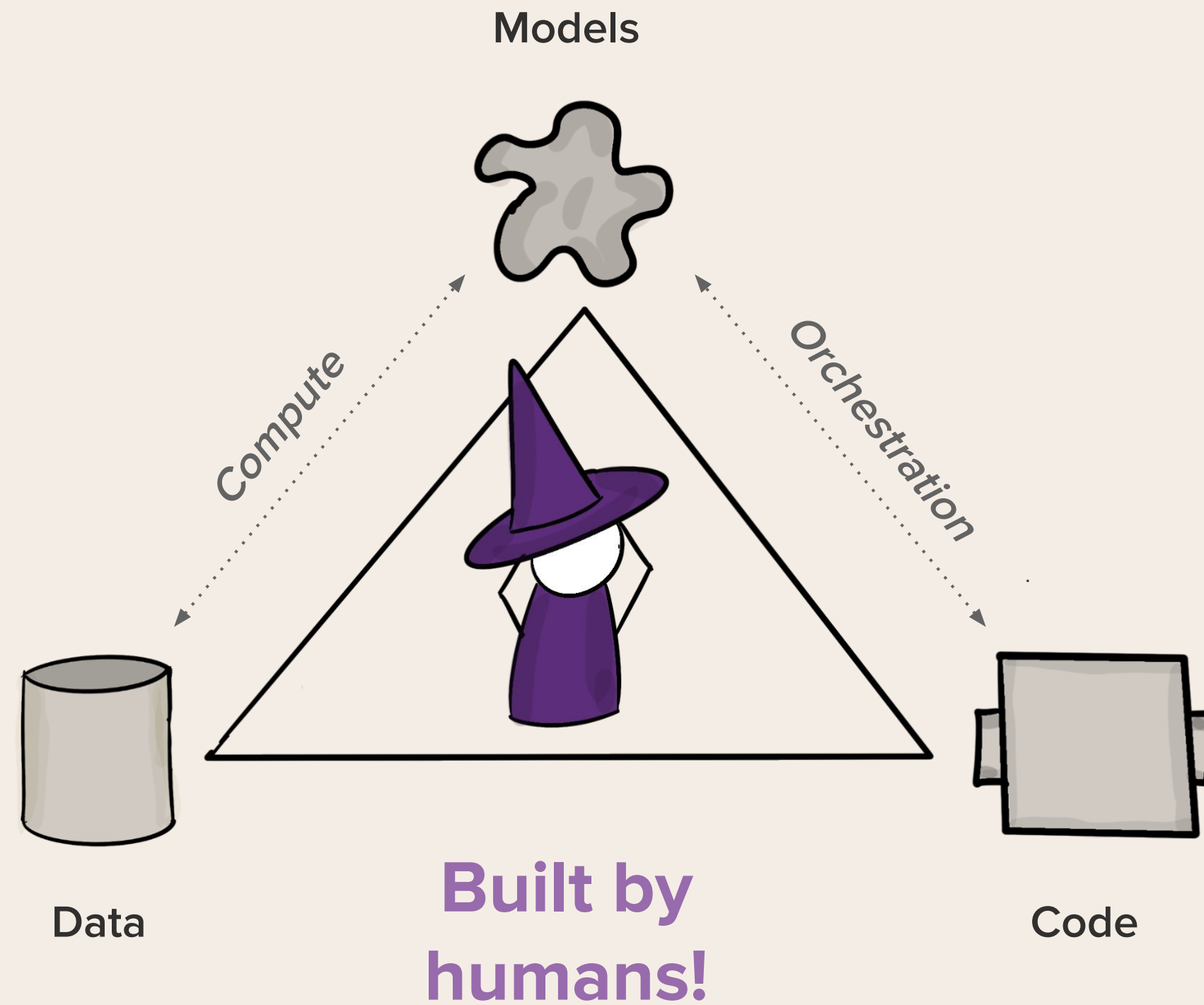
Building Blocks of ML Systems



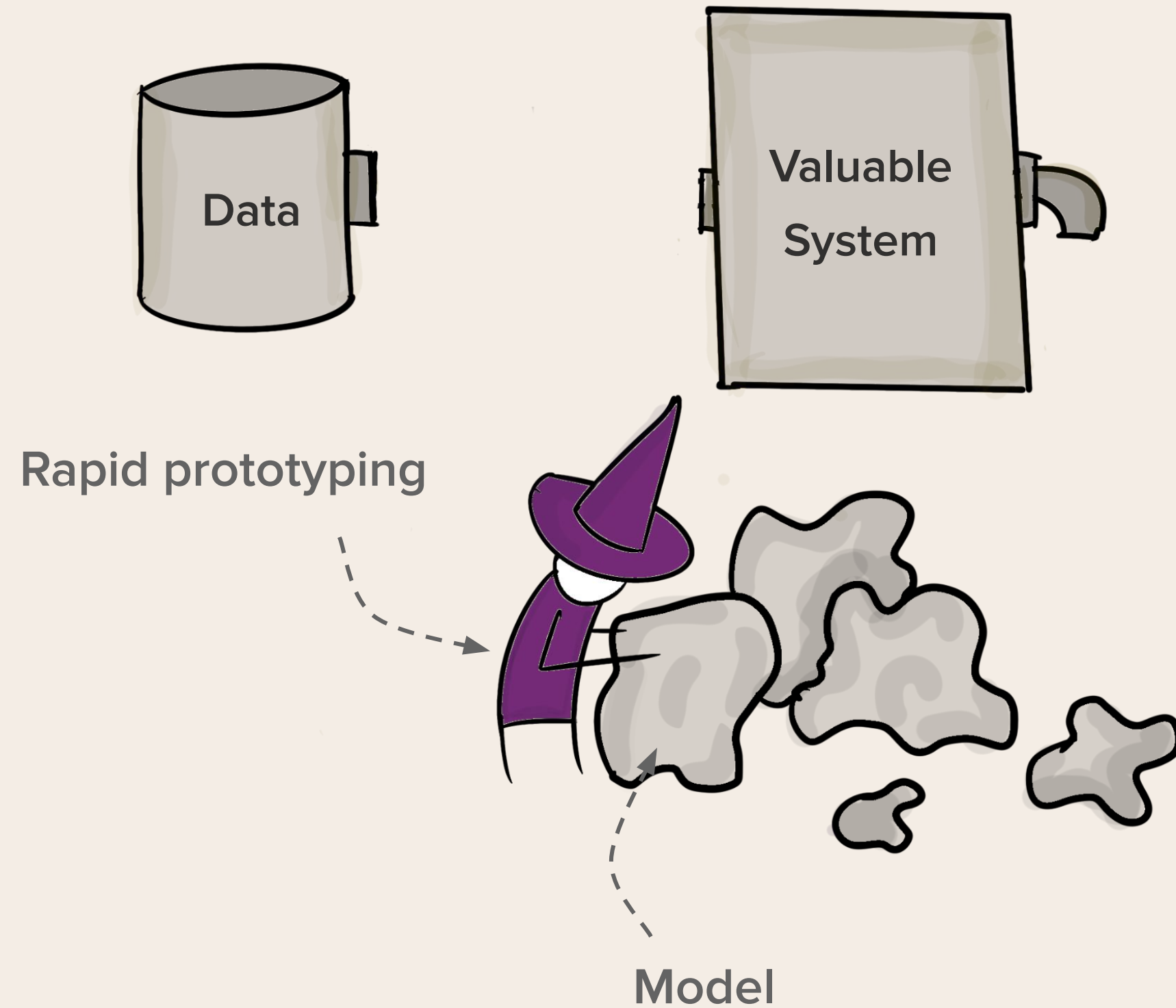
Building Blocks of ML Systems



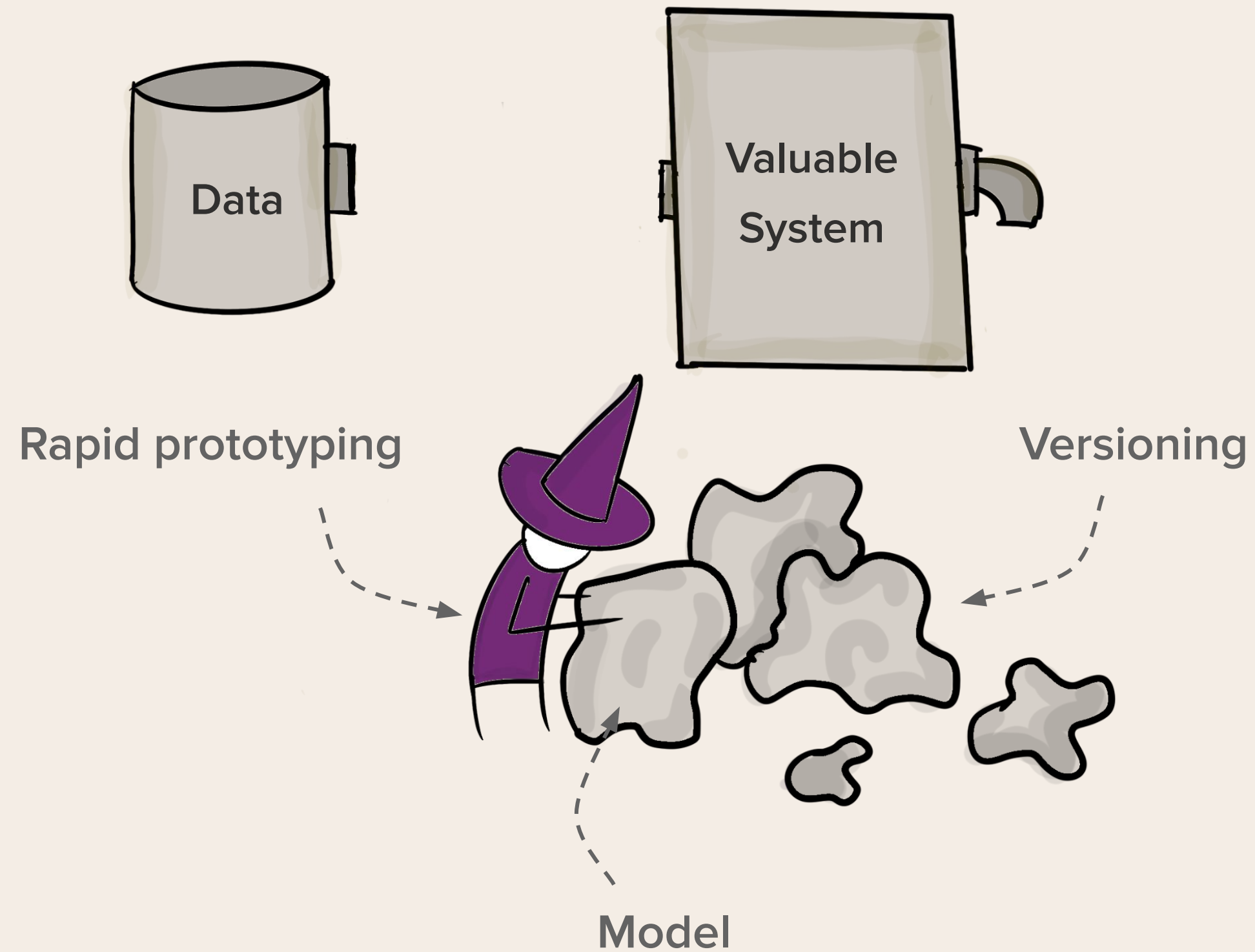
Building Blocks of ML Systems



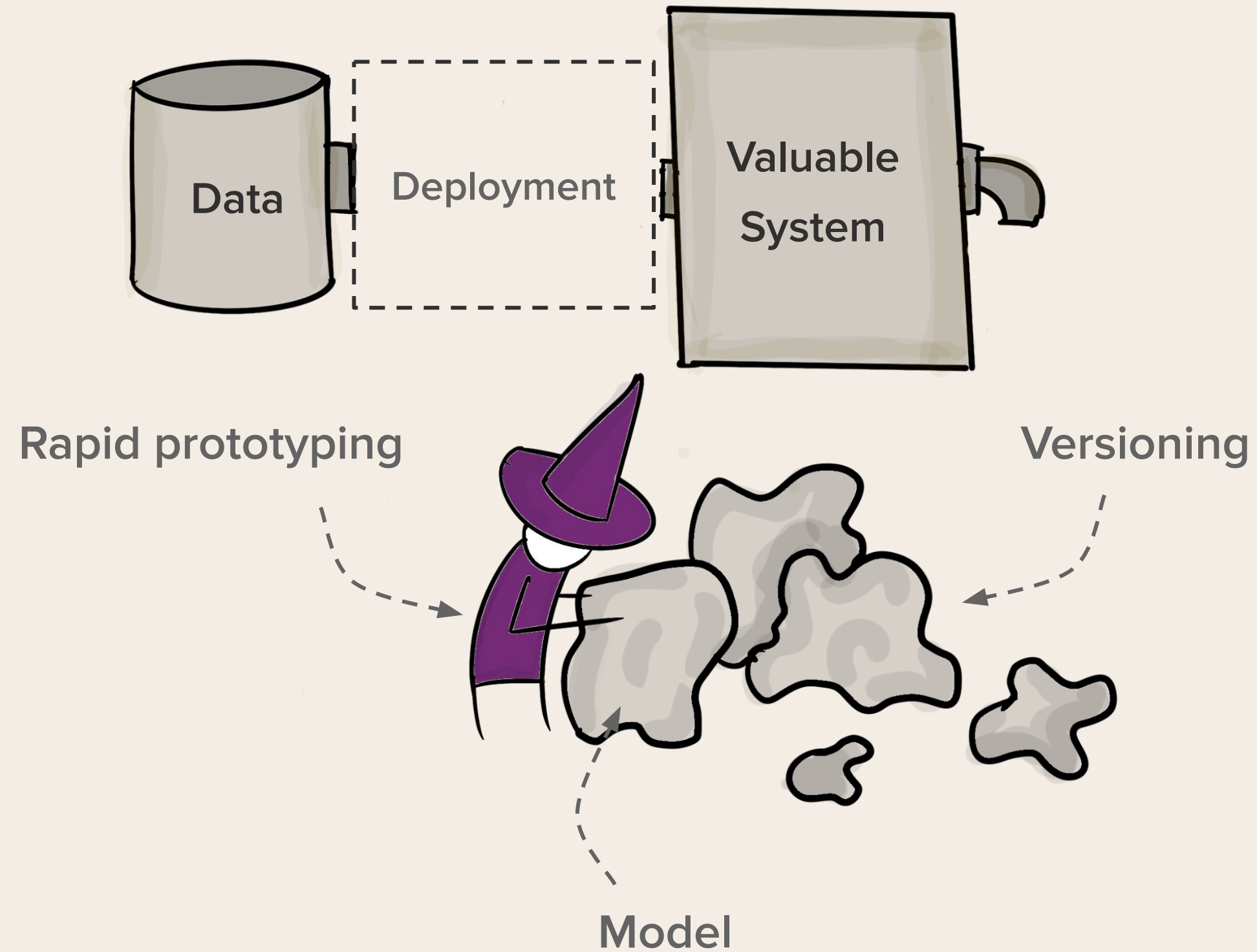
ML systems are built *iteratively* by human beings



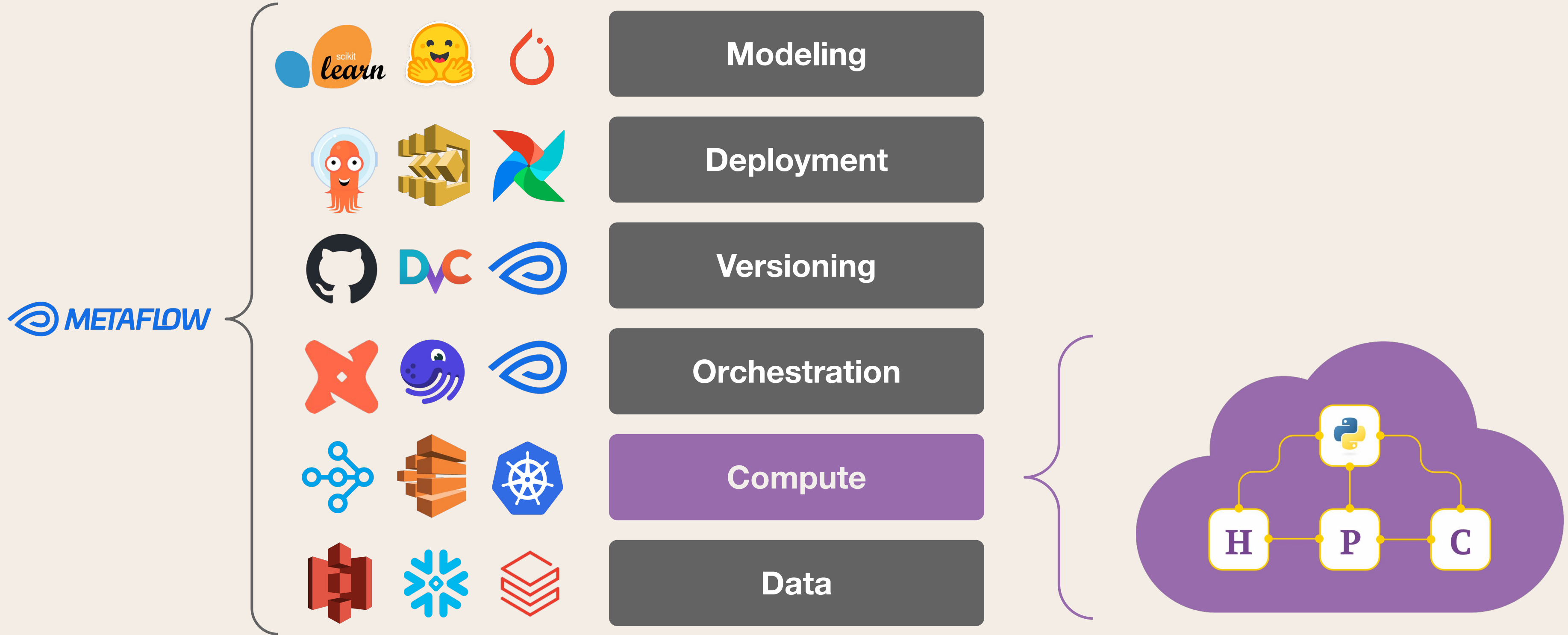
ML systems are built *iteratively* by human beings



ML systems are built *iteratively* by human beings v



Researchers need infrastructure for building AI systems





Demo time!

<https://github.com/outerbounds/pydata-global-2023-hpc/tree/main/torchrun-cluster>

Recap

- Cloud is here to stay
- AI is here to stay
- Competitive model sizes are growing fast, bigger models need bigger compute
- AI compute primitives build on established HPC tech
- It is important to make HPC primitives more accessible to data scientists who think in Python

Outerbounds

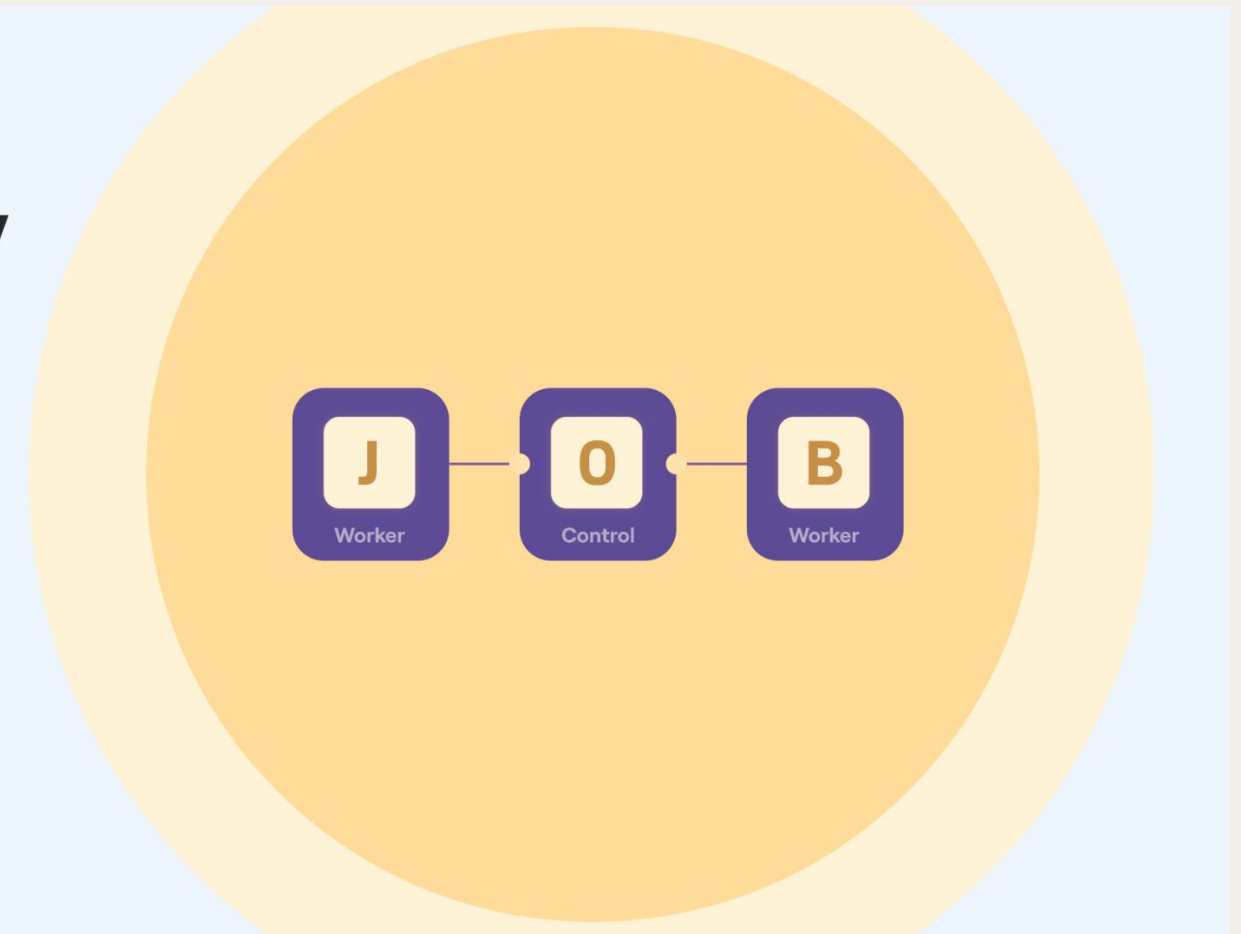
Compute with Metaflow



<https://outerbounds.com/blog/compute-with-metaflow/>

Metaflow

@metaflow_ray
@deepspeed
@tensorflow
@torchrun
@mpi



<https://outerbounds.com/blog/distributed-training-with-metaflow/>

More HPC in the cloud resources

Slurm

- [Introduction to Slurm](#) YouTube Series
- [Slurm database usage](#) YouTube Series
- [Documentation](#)

Kubernetes

- [Documentation](#)
- [Kubernetes Meets High-Performance Computing](#) Blog

General

- [Design Considerations for Building and Running Containerized MPI Applications](#) Paper
- [HPC on the Cloud: Slurm Cluster vs Kubernetes](#) Blog
- [Hacker news thread](#) on Slurm, Kubernetes, Condor, and more!

More CSP resources

AWS

- [HPC Tech Shorts](#) YouTube Series
- [Optimizing your AWS Batch architecture for scale with observability dashboards](#) Blog
- [Elastic Kubernetes Service \(EKS\)](#) Documentation
- [Distributed training in Amazon SageMaker](#) Documentation

GCP

- [Batch](#) Documentation
- [Google Kubernetes Engine \(GKE\)](#) Documentation
- [Cloud HPC Toolkit](#) YouTube Series

Azure

- [Batch](#) Documentation
- [Azure Kubernetes Service \(AKS\)](#) Documentation

More Metaflow resources

Metaflow Documentation

<https://outerbounds.com>

Outerbounds CEO, Ville Tuulos, and
Head of DevRel, Hugo Bowne-Anderson
are giving talks at PyData global too!

Join us for more about how compute relates to an AI systems point-of-view.

Compute Anything with Metaflow, today at 12:30 PST

Full-stack Machine Learning and Generative AI for Data Scientists, today at 1 PST

Thanks!

Join Metaflow Community

With over 3000 data scientists and engineers at

<http://slack.outerbounds.co>



Feel free to DM me at

@eddie 🤗

DIY path: Kubernetes vs. Slurm, or is it both...?

Introducing SUNK: A Slurm on Kubernetes Implementation for HPC and Large Scale AI

<https://www.coreweave.com/blog/sunk-slurm-on-kubernetes-implementations>



DIY path

A drastically over-simplified decision tree

- if massive scale distributed training & don't want to customize Kubernetes → Slurm
- else if already using Slurm and like it → Slurm
- else → Kubernetes

Argument for batch jobs on Kubernetes

1. Kubernetes is flexible.
2. Kubernetes has developer ecosystem momentum.
3. Because of 1 and 2, many custom Kubernetes resources are emerging for HPC-style batch job queues.
4. Most cloud companies already maintain a pool of compute managed by a Kubernetes cluster.
5. Because of 3 and 4, it is easier and more flexible to enable AI researchers on Kubernetes.