# SPARK+AI SUMMIT 2020

Organized by databricks

# Ray: Enterprise-Grade, Distributed Python

Dean Wampler

Anyscale

# Feedback

Your feedback is important to us.

## Don’t forget to rate and review the sessions.

SPARK+AI SUMMIT

#Datateams #SparkAISummit

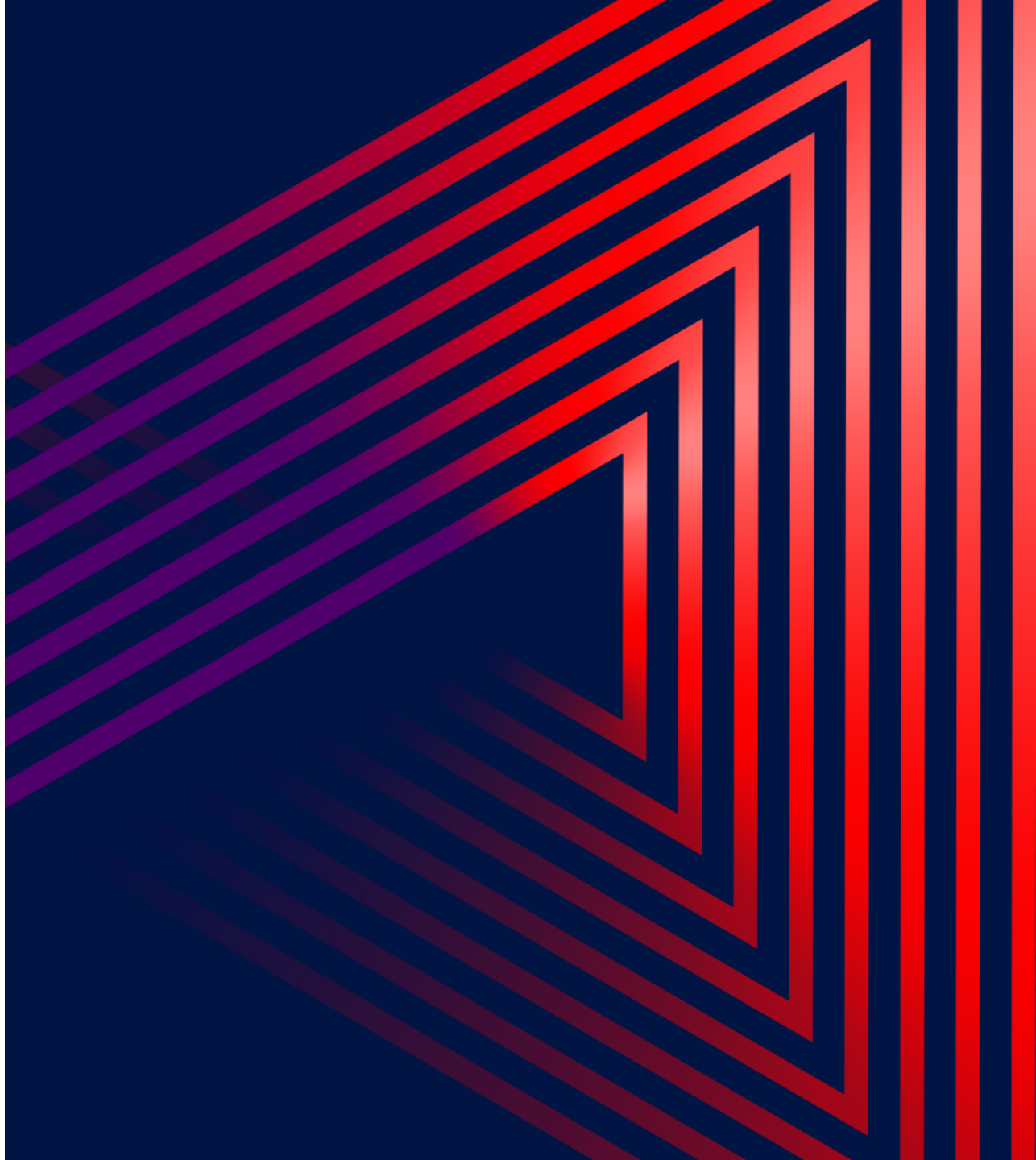
# Agenda

Why Ray?

Demo

When to use Spark? When to use Ray?

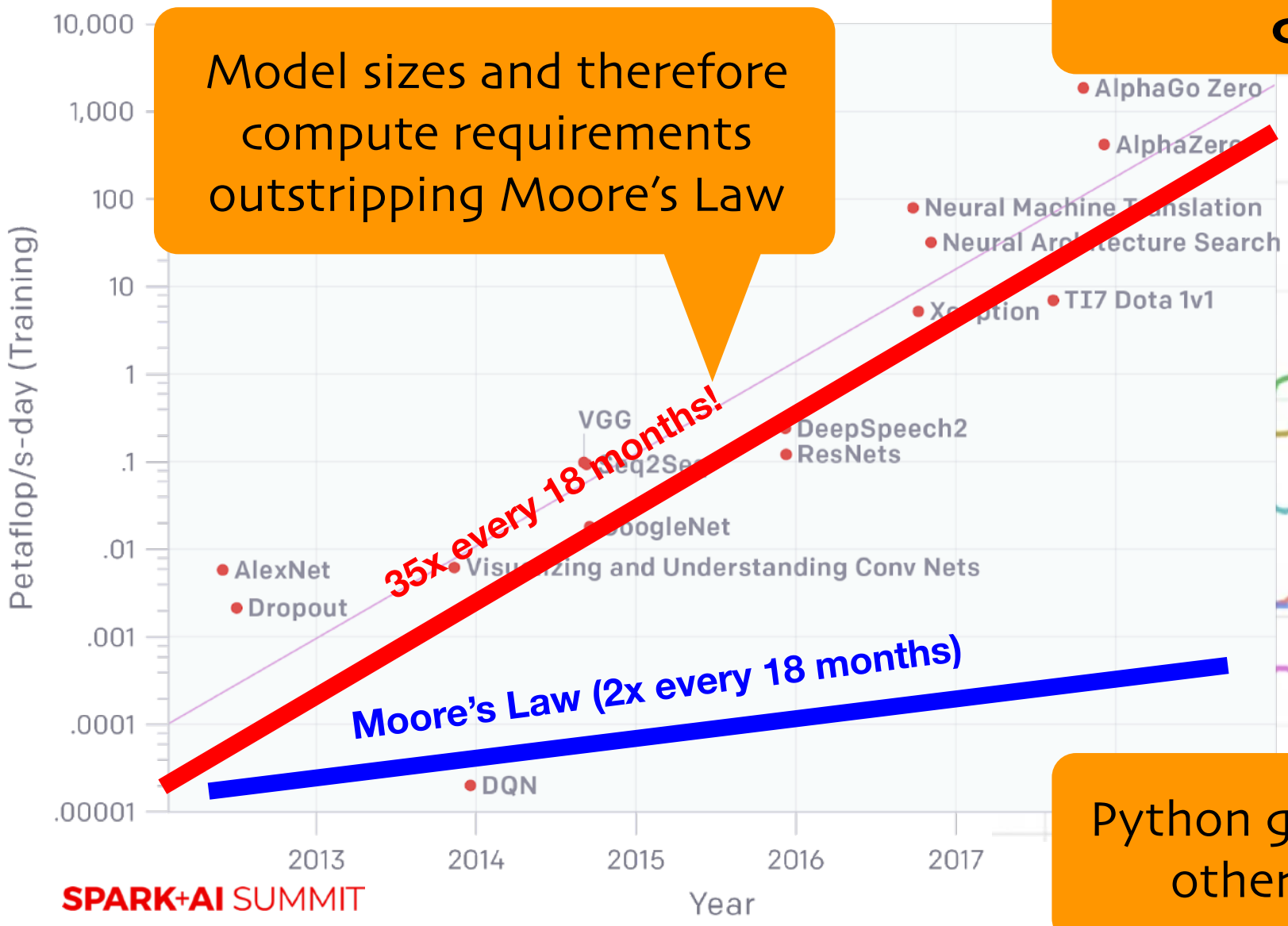
How to get started with Ray

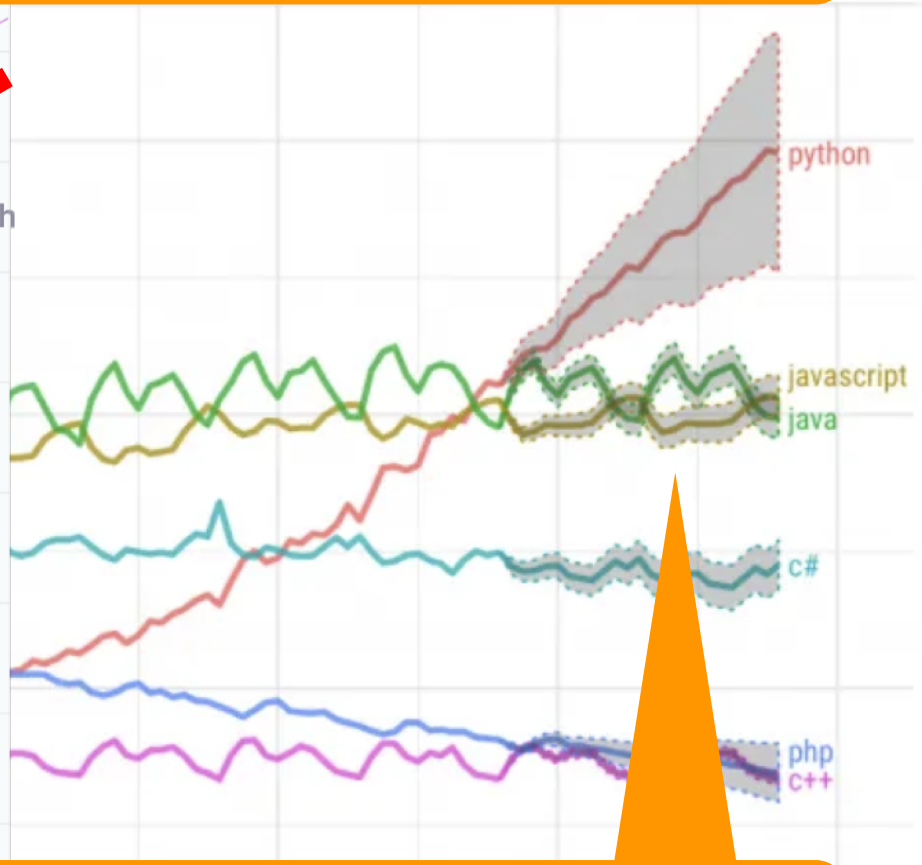


# Why Ray?

Hence, there is a pressing need for **robust, easy to use solutions for distributed Python**

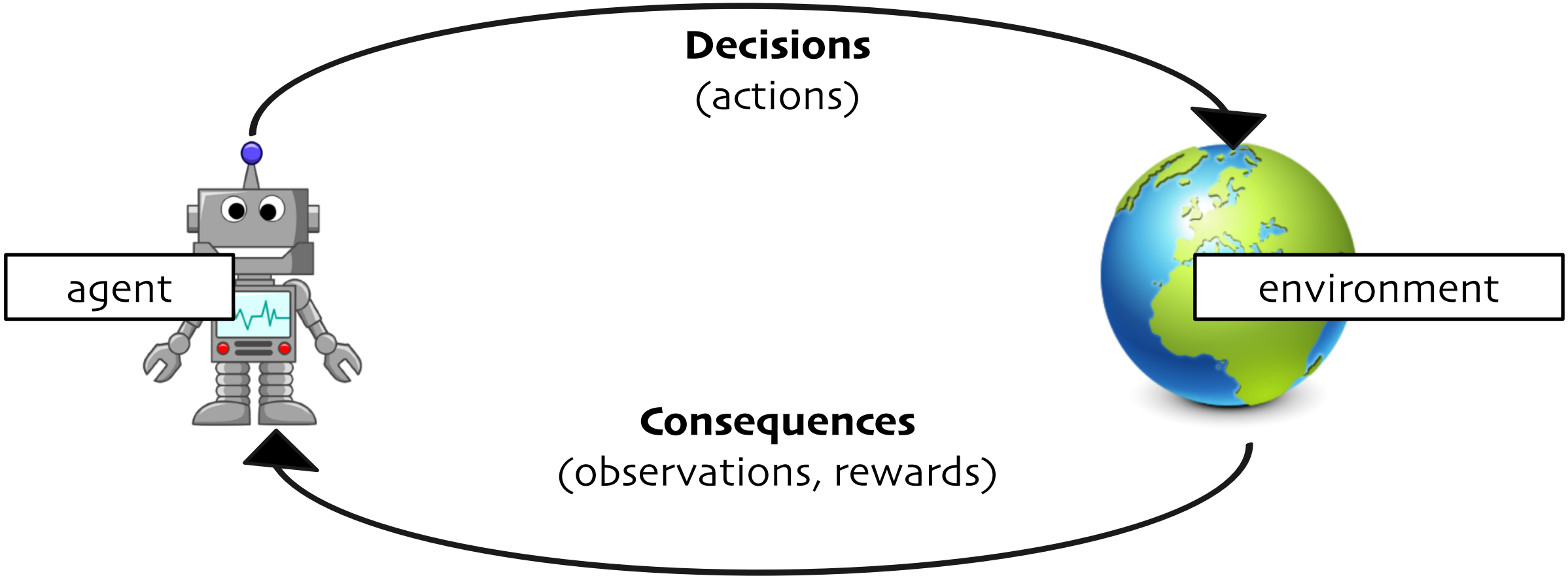
Model sizes and therefore compute requirements outstripping Moore's Law



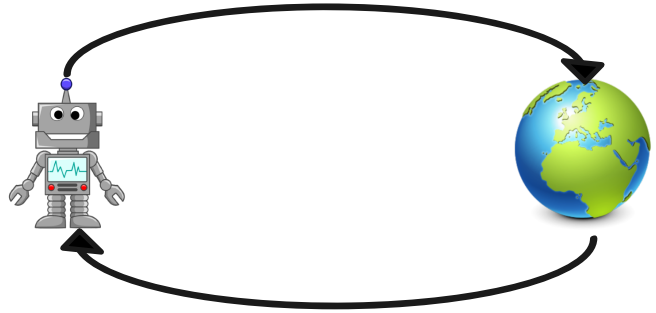



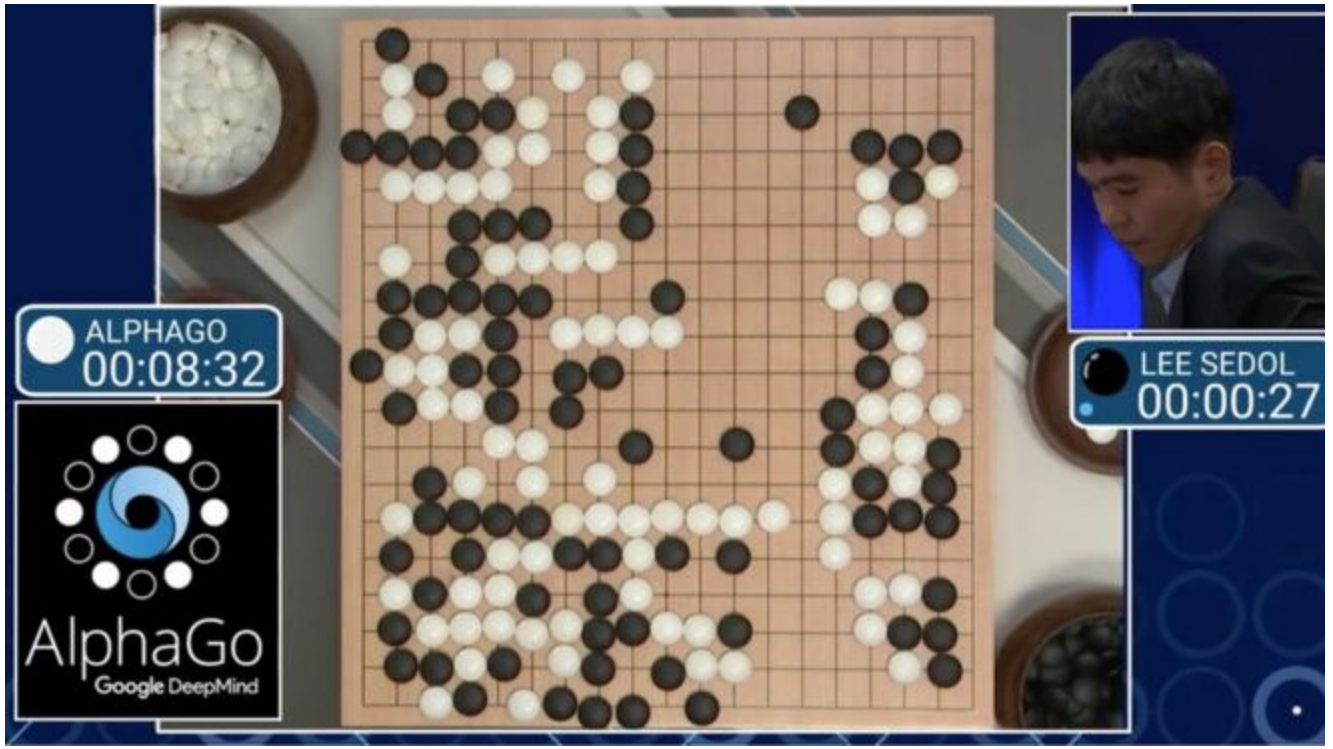

Python growth driven by ML/AI and other data science workloads

# Reinforcement Learning: Motivation for Ray


Decisions
(actions)
agent
environment
Consequences
(observations, rewards)

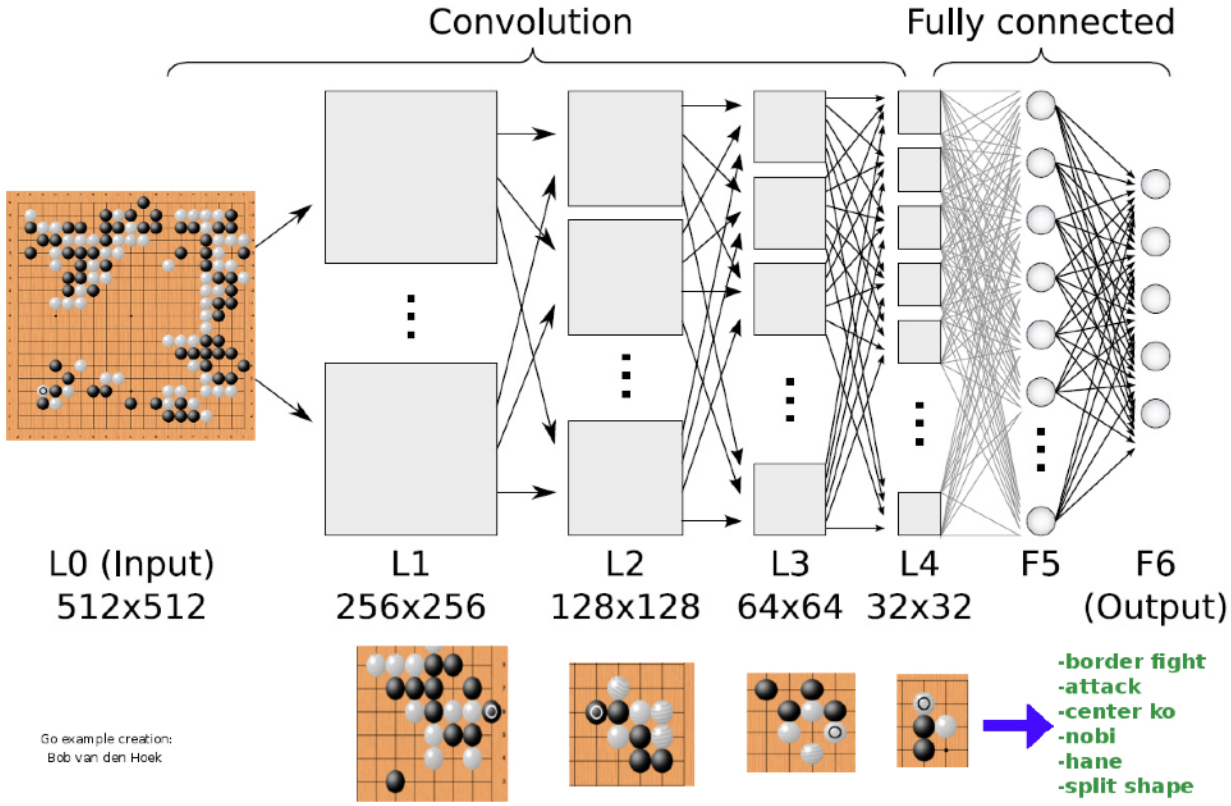
# Beating Lee Sedol…

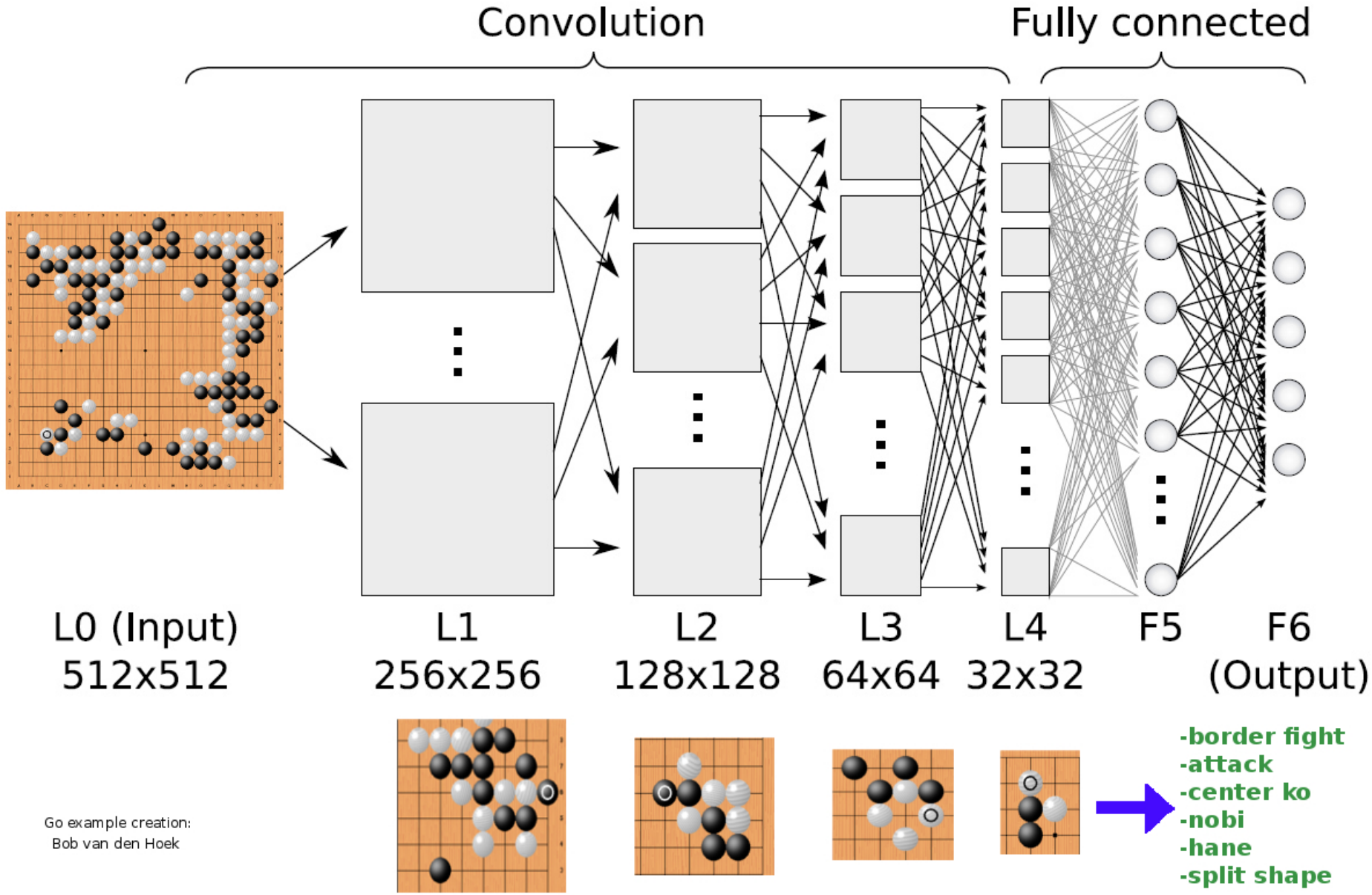




# AlphaGo (Silver et al. 2016)

- Observations
  - Board state
- Actions
  - Where to place stones
- Rewards
  - 1 if win
  - 0 otherwise



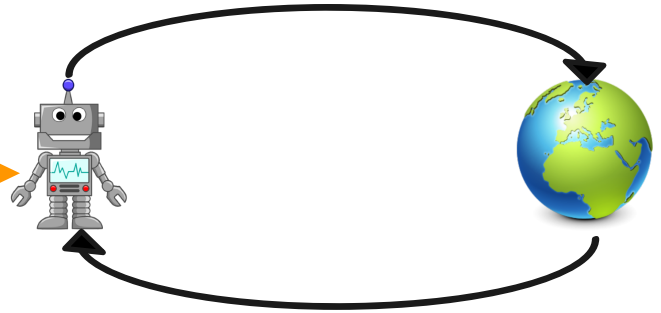

Convolution
Fully connected
L0 (Input)
512x512
L1
256x256
L2
128x128
L3
64x64
L4
32x32
F5
F6
(Output)
Go example creation:
Bob van den Hoek
-border fight
-attack
-center ko
-nobi
-hane
-split shape

# Diverse Compute Requirements Motivated Creation of Ray!

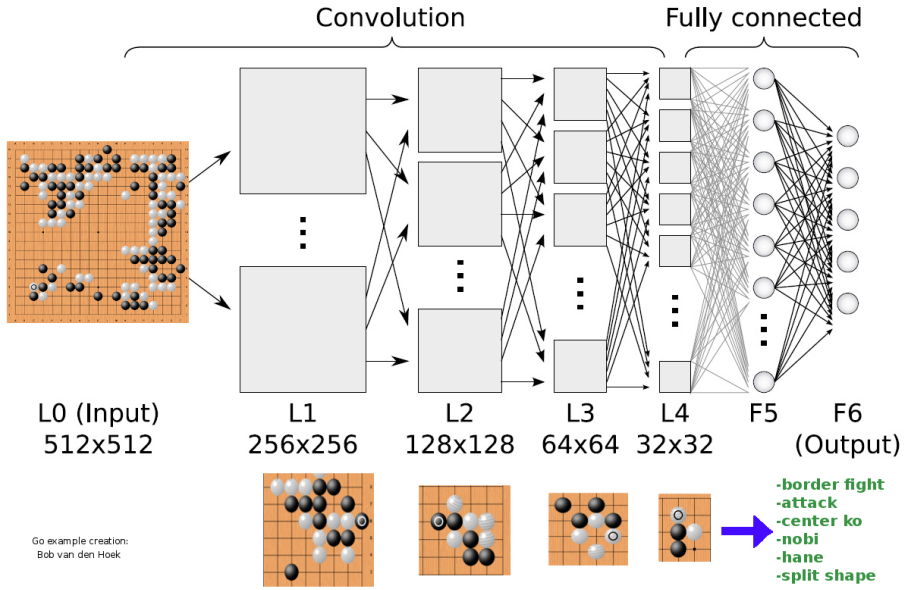
And repeated play, over and over again, to train for achieving the best reward

Neural network "stuff"

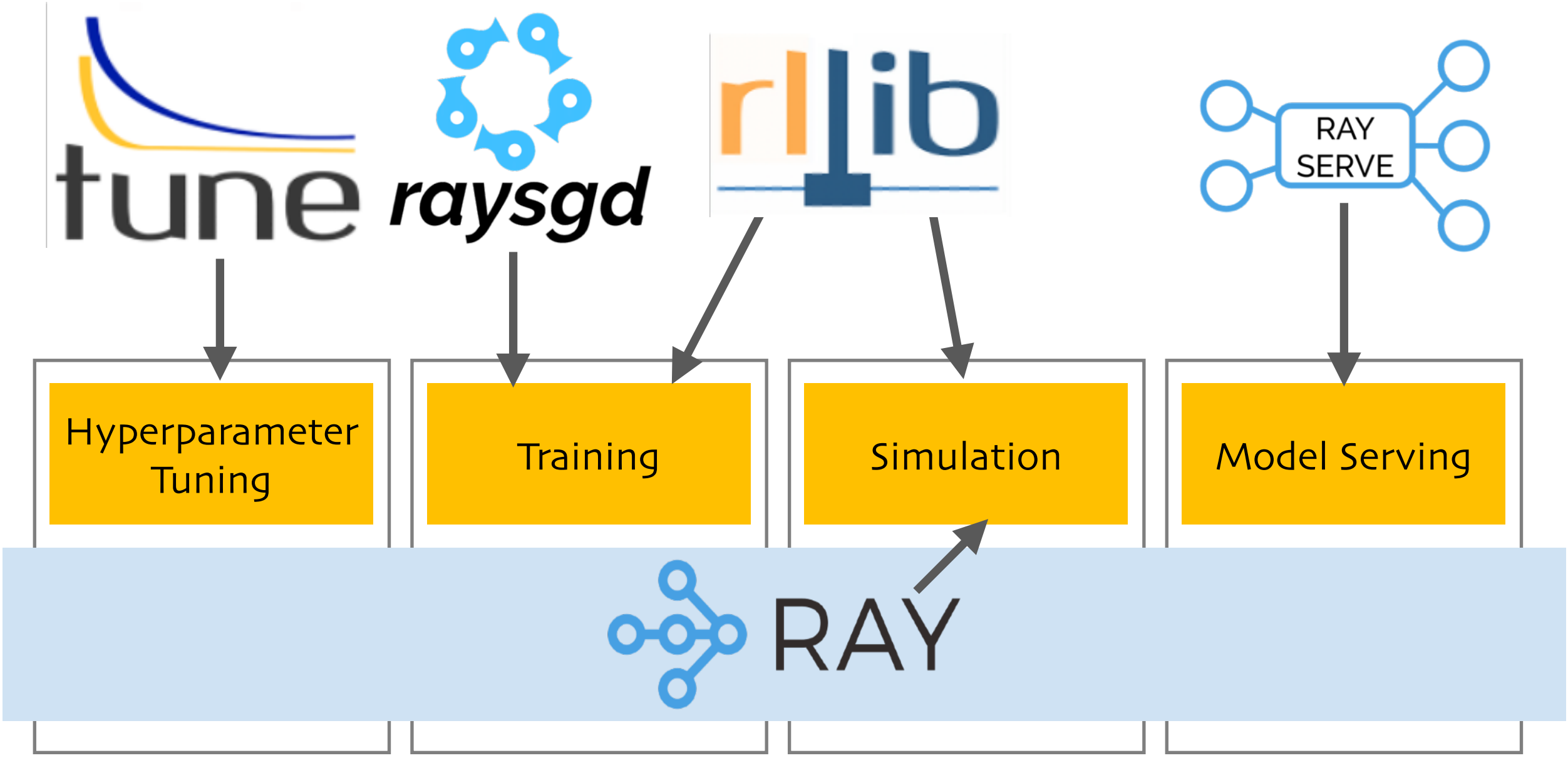
Simulator (game engine, robot sim, factory floor sim…)



Complex agent?

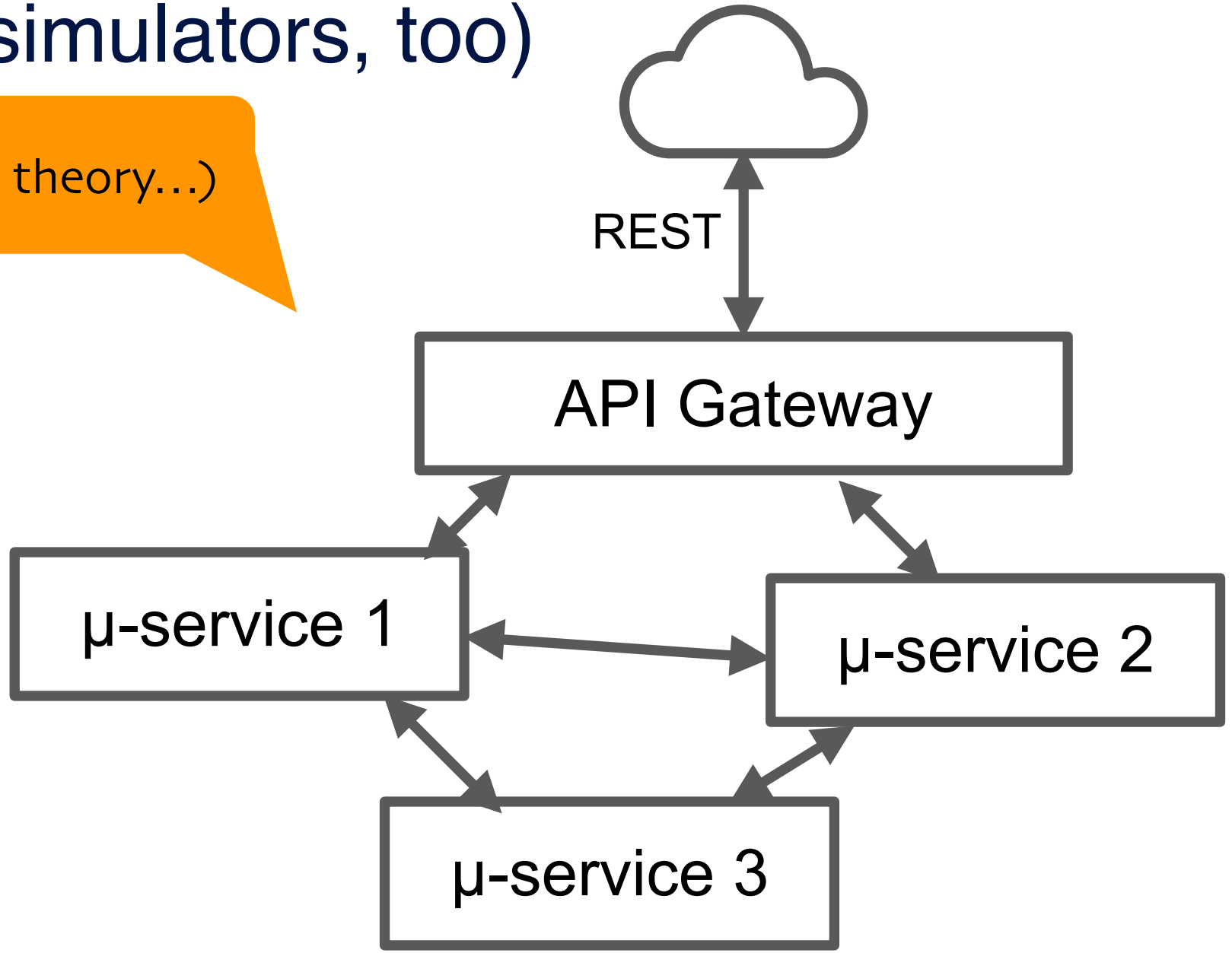


# The Ray Ecosystem


tune
raysgd
rllib
RAY
SERVE
Hyperparameter Tuning
Training
Simulation
Model Serving
RAY

# Microservices (simulators, too)

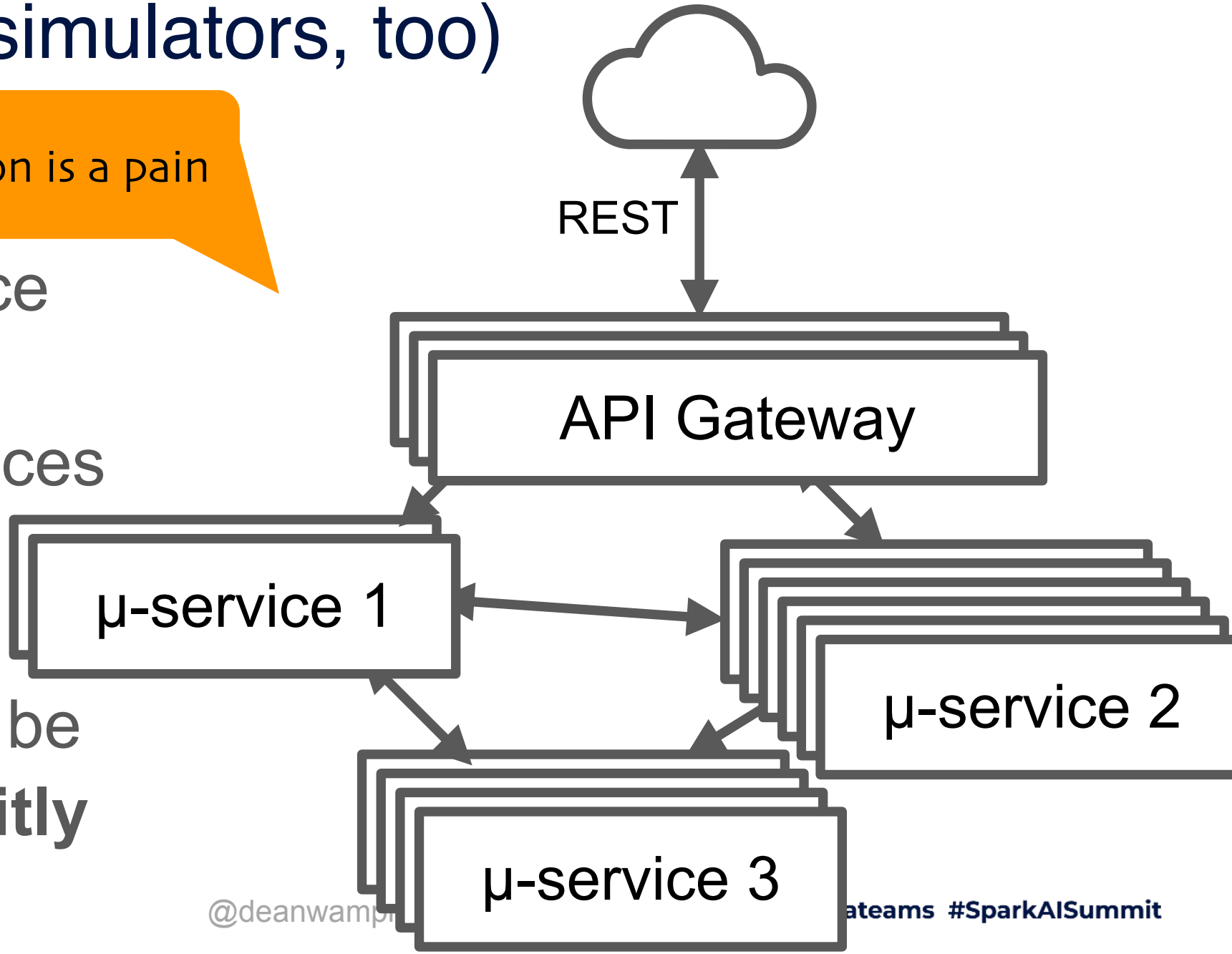
Nice! (In theory…)



# Microservices (simulators, too)

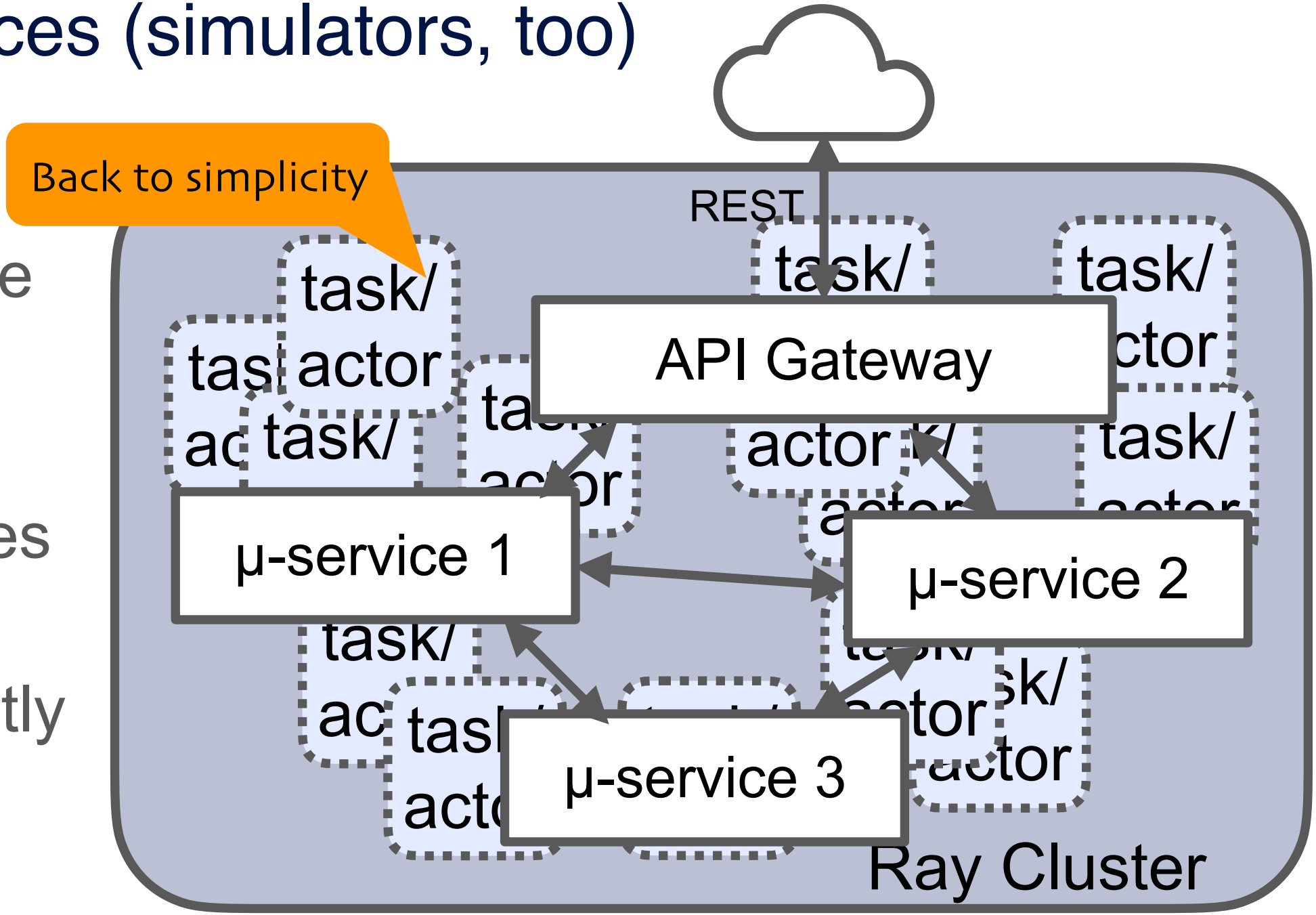
Production is a pain

- Each microservice has a different number of instances for scalability & resiliency
- But they have to be managed **explicitly**



# Microservices (simulators, too)

- Back to one "logical" instance
- Ray handles scaling transparently



# Demo!

# When to use Spark
# When to use Ray

# Where Spark Excels

- Massive-scale data sets
  - Uniform, records with a schema
  - Efficient, parallelized transformations
  - SQL
- Batch analytics
- Stream processing
- Intuitive, high-level abstractions for data science & engineering tasks

# Where Ray Excels

- Highly non-uniform data graphs
  - Think typical “in-memory object models”, but distributed
  - Handles distributed state intuitively
- Highly non-uniform task graphs
  - Small to large scale tasks
- Intuitive API for the “90%” of cases
- Supports compute problems ranging from
  - general services, games, and simulators
- to
  - stochastic gradient descent, HPO, …

# Getting started with Ray

# If you're already using these…

- asyncio
- joblib
- multiprocessing.Pool

For example, from this:

```
from multiprocessing.pool import Pool
```

To this:

```
from ray.util.multiprocessing.pool import Pool
```

- Use Ray's implementations
  - Drop-in replacements
  - Change import statements
  - Break the one-node limitation!

# Ray Community and Resources

- ray.io - entry point for all things Ray
- Tutorials: anyscale.com/academy
- github.com/ray-project/ray
- anyscale.com/events

# Feedback

Your feedback is important to us.

## Don’t forget to rate and review the sessions.

SPARK+AI SUMMIT

#Datateams #SparkAISummit

# SPARK+AI SUMMIT 2020

Organized by databricks