Systems plus ML:
When the sum is greater than its parts

Ion Stoica

UC Berkeley, Director of RISELab

November 5, 2019

riselab
UC Berkeley

Studies the design of

real-time,

intelligent,

secure, and

explainable

algorithms and systems

riselab
UC Berkeley

Studies the design of

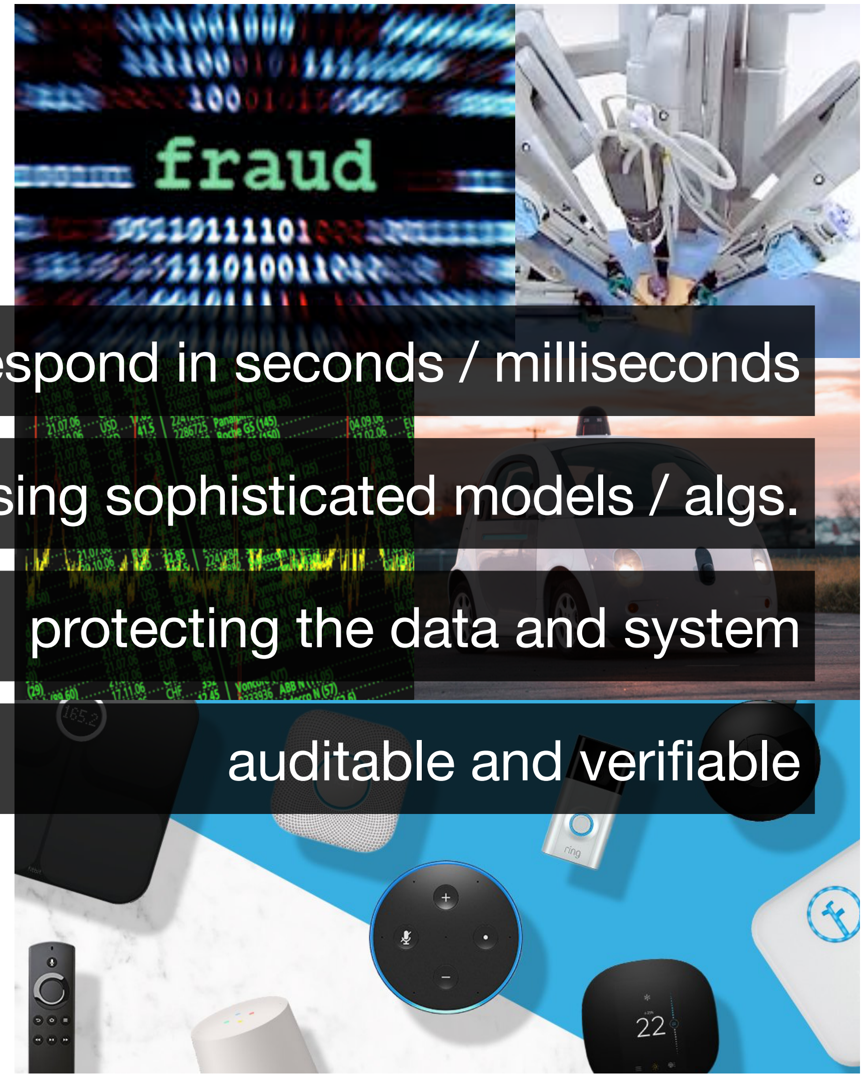
real-time,

intelligent,

secure, and

explainable

algorithms and systems



respond in seconds / milliseconds

using sophisticated models / algs.

protecting the data and system

auditable and verifiable

riselab
UC Berkeley

Studies the design of

real-time,

intelligent,

secure, and

explainable

algorithms and systems

Interdisciplinary Lab

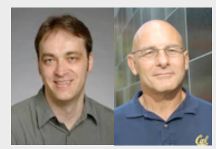
AI

Abbeel Goldberg Gonzalez Jordan Mahoney

Security

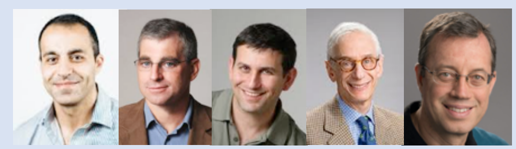
Joseph Popa Song

Hardware



Asanovic Patterson

Systems



Ghodsi Culler Hellerstein Katz Stoica

Collaborate with:

NSF

Alibaba.com, amazon web services, 蚂蚁金服 ANT FINANCIAL, arm, CapitalOne, ERICSSON, facebook, Google, intel, Microsoft, NVIDIA, Scotiabank, splunk>, vmware

Systems ML

Positive feedback loop to accelerate ML and Systems

Systems $\rightarrow$ ML

- Ray

Systems $\leftarrow$ ML

- Deep queries
- Neurocuts
- Autopandas

Systems **ML**

- **Ray**

Systems ML

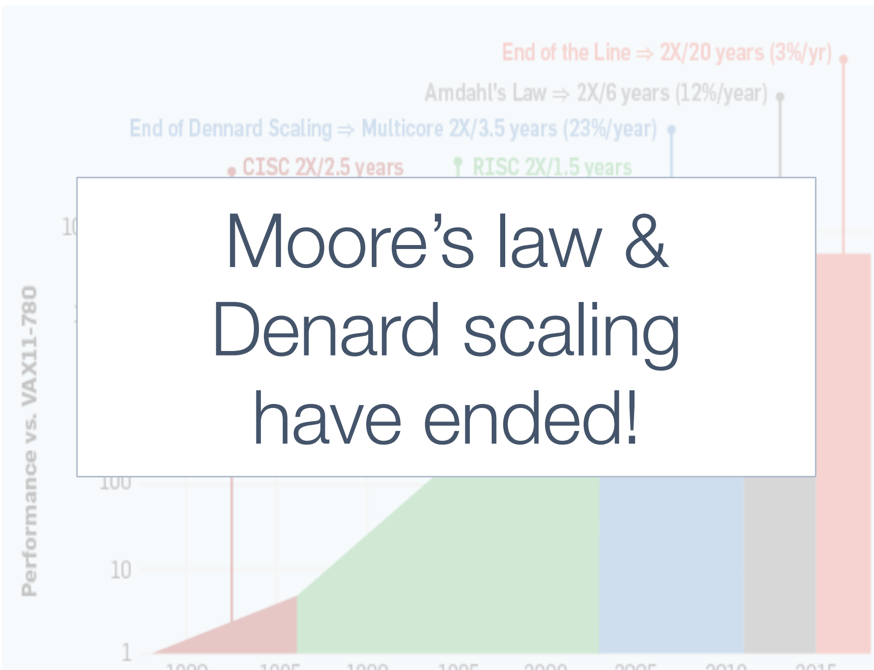
- Deep queries
- Neurocuts
- Autopandas

Trends

Apps becoming **distributed**

Apps becoming more **complex**

Apps becoming **distributed**



Moore's law & Denard scaling have ended!

Training largest models: doubles every 3.5 months (**35x** over 18 months) !

No choice but to distribute apps

Apps becoming more **complex**

Virtually all apps will become AI centric

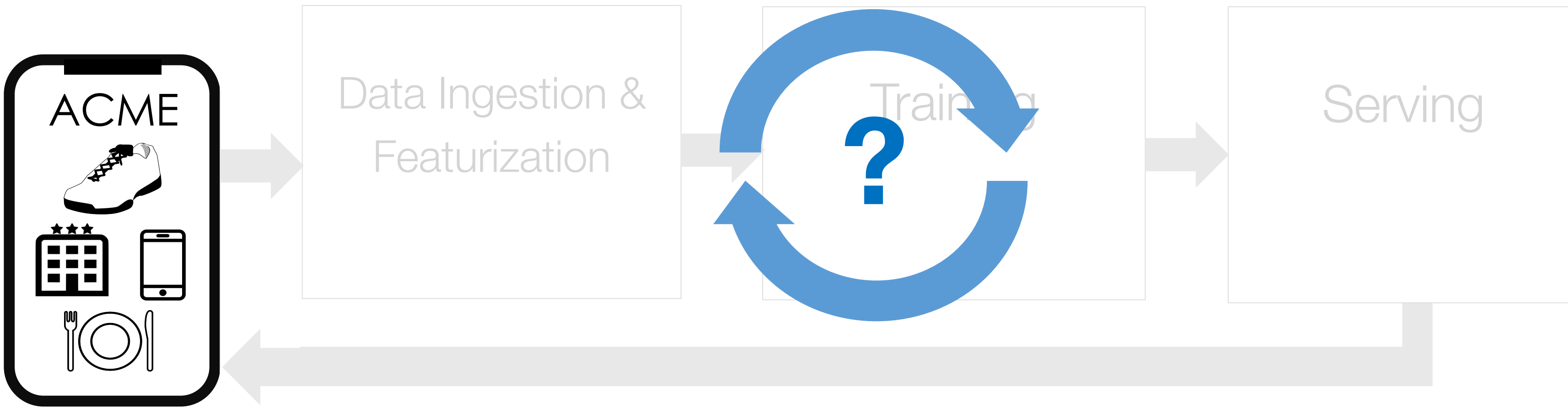
Example: In-app promotion

A **real** use case:

- Recommend services, products
- Largest fintech company in the world

Example: In-app promotion

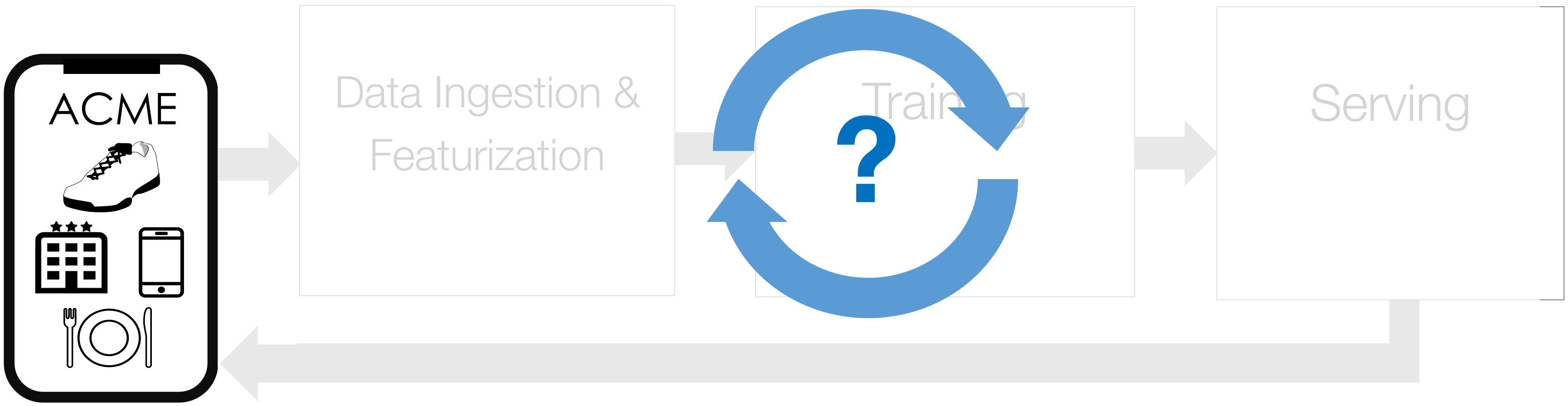
Example: In-app promotion



Two questions:

- How fast can we update the model?
- How much does it matter?

Example: In-app promotion



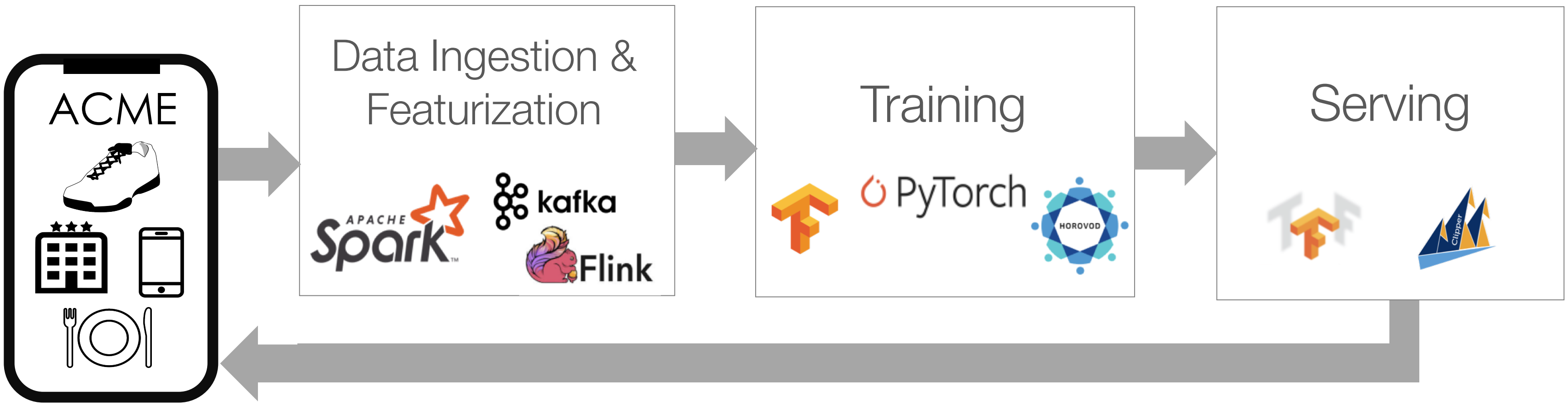
Model updated every **1 day**

+ 5% CTR (Click Through Rate)

Model updated every **1 hour** (state-of-the-art solution)

Want to get lower, but how?

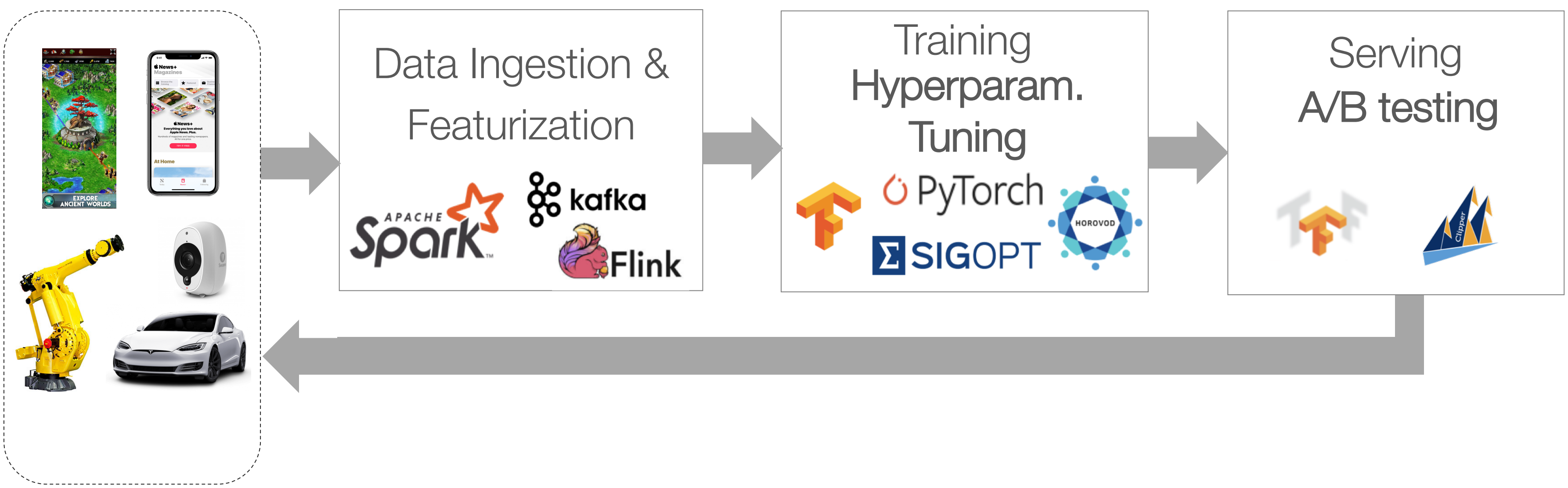
Example: In-app promotion



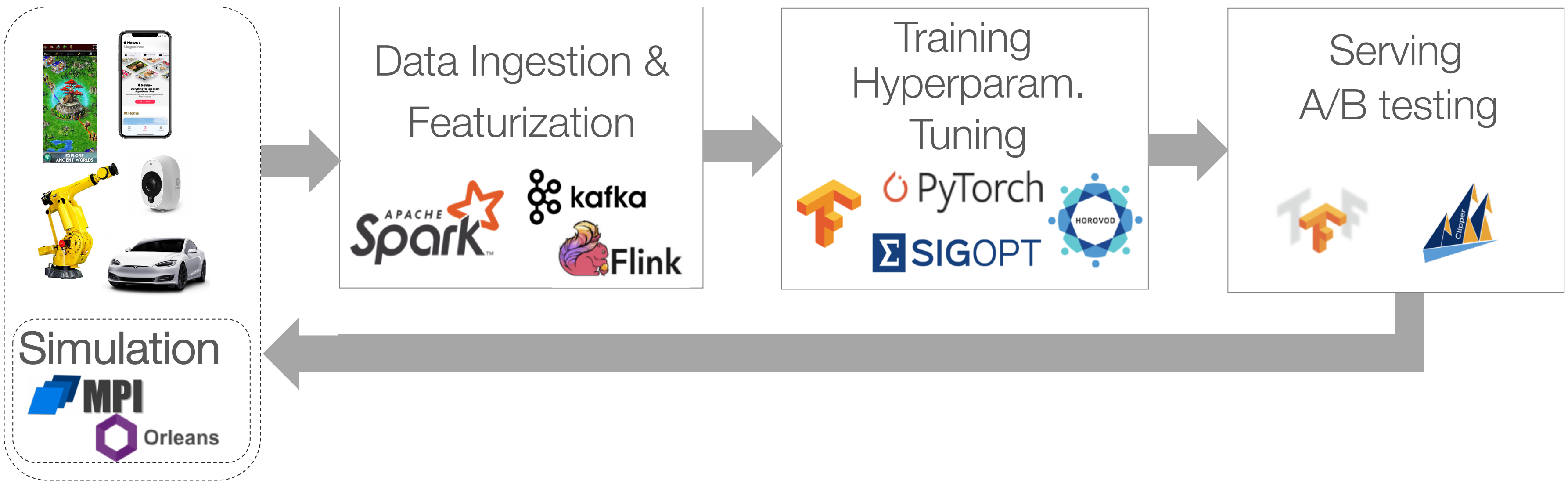
Previous solution: integrate best-of-breed frameworks

Challenges: end-to-end delay, development, management cost

Even more complex patterns!



Even more complex patterns!



Reinforcement Learning

Today's ML Ecosystem

RAY

Distributed systems	Distributed systems	Distributed systems	Distributed systems	Distributed systems	Distributed systems
Training	Model Serving	Hyperparam. Tuning	Streaming	Simulation	Featurization
TensorFlow, Horovod, PyTorch, scikit-learn	TF, Clipper	SigOpt, talos	Apache Spark, Kafka, Flink	MPI, Orleans	Apache Spark, Hadoop

RAY *

Libraries

Training	Model Serving	Hyperparam. Tuning	Streaming	Simulation	Featurization

RAY

General-purpose distributed computing framework for Python (and Java)

* "Ray: A Distributed Framework for Emerging AI Applications", Philipp Moritz et al, OSDI 2018

Function

```
def read_array(file):
    # read ndarray “a”
    # from “file”
    return a


def add(a, b):
    return np.add(a, b)

a = read_array(file1)
b = read_array(file2)
sum = add(a, b)
```

Object

```
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter()
c.inc()
c.inc()
```

RAY

Function → Task

```python
@ray.remote
def read_array(file):
    # read ndarray “a”
    # from “file”
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

a = read_array(file1)
b = read_array(file2)
sum = add(a, b)
```

Object → Actor

```python
@ray.remote
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter()
c.inc()
c.inc()
```

Function → Task

```python
@ray.remote
def read_array(file):
    # read ndarray “a”
    # from “file”
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

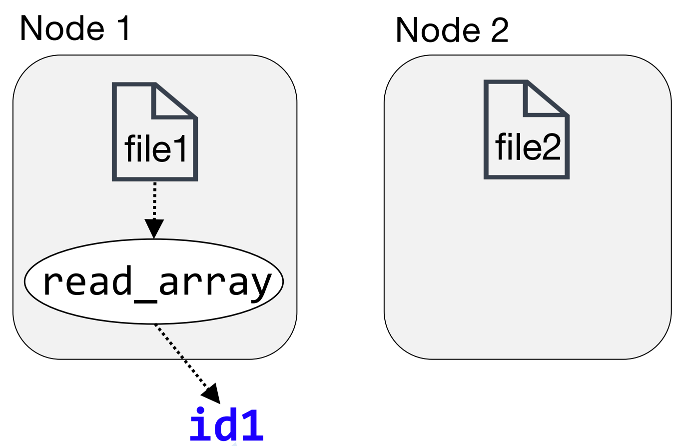
Object → Actor

```python
@ray.remote
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter.remote()
id4 = c.inc.remote()
id5 = c.inc.remote()
```

Task API

```
@ray.remote
def read_array(file):
    # read ndarray “a”
    # from “file”
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

- Blue variables are Object IDs
- Similar to futures

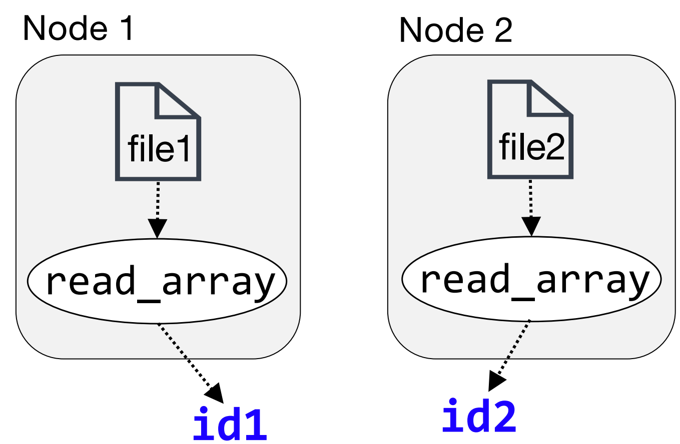


Return id1 immediately, before `read_array()` finishes

Task API

```
@ray.remote
def read_array(file):
    # read ndarray "a"
    # from "file"
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

- Blue variables are Object IDs
- Similar to futures

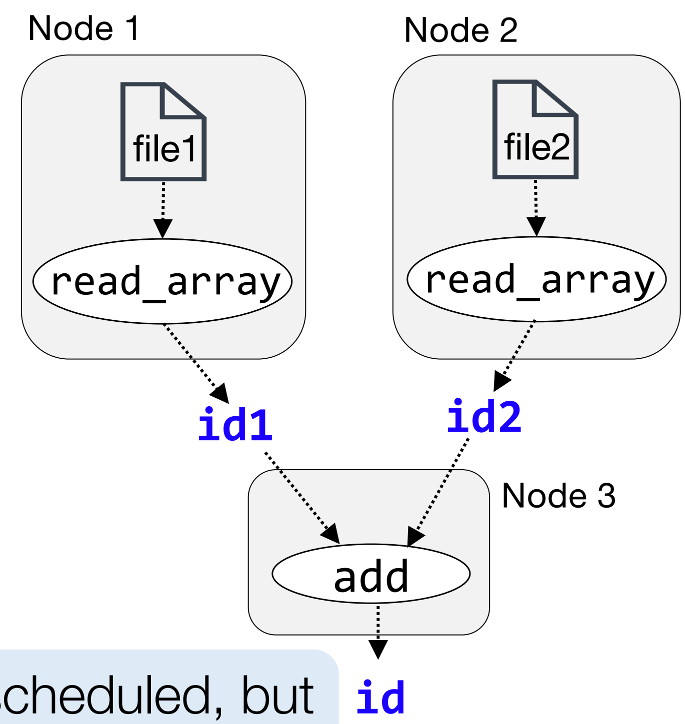


Dynamic task graph:
build at runtime

Task API

```python
@ray.remote
def read_array(file):
    # read ndarray “a”
    # from “file”
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

- Blue variables are Object IDs
- Similar to futures



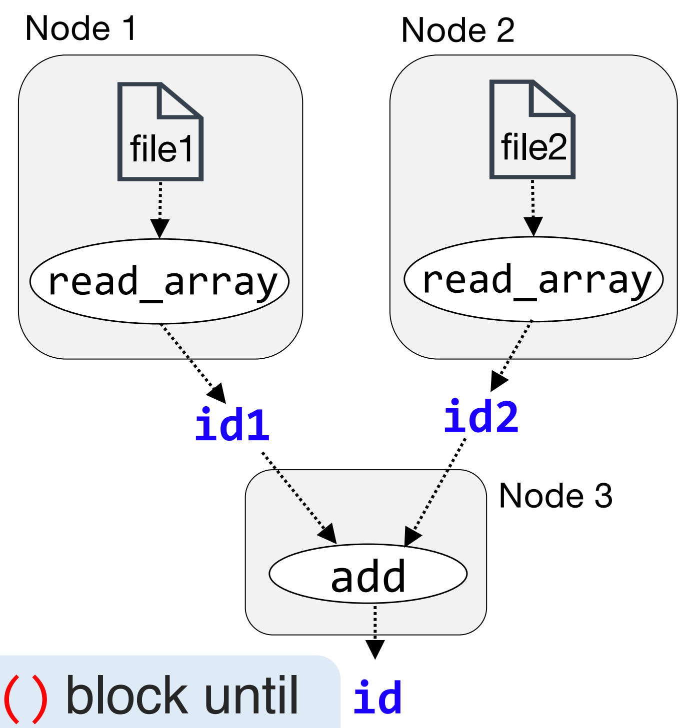
Every task scheduled, but not finished yet

Task API

```
@ray.remote
def read_array(file):
    # read ndarray "a"
    # from "file"
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

ray.get() block until result available

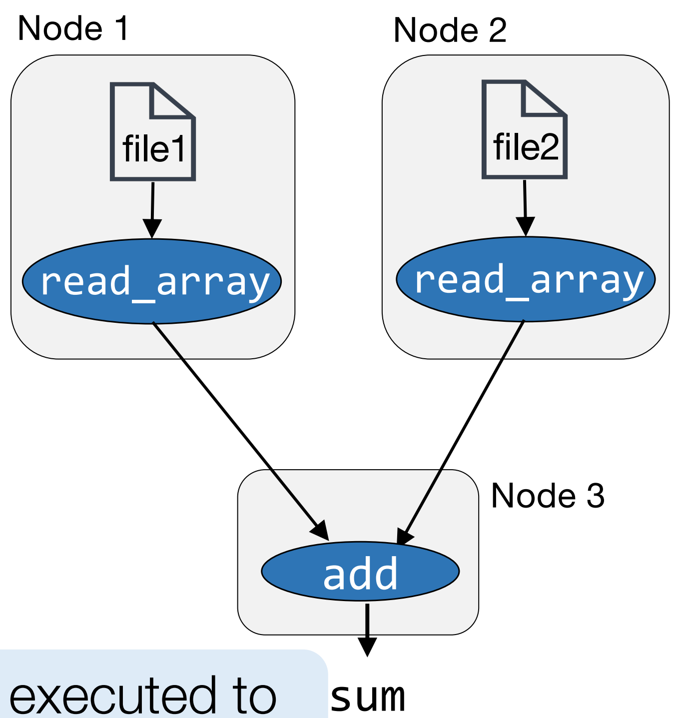
- Blue variables are Object IDs
- Similar to futures



Task API

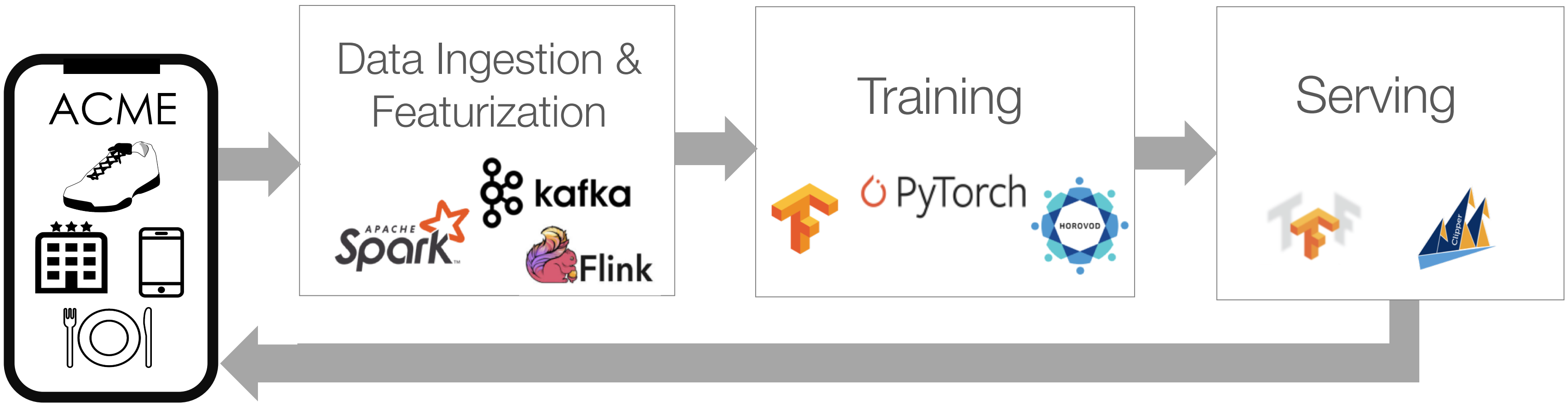
```
@ray.remote
def read_array(file):
    # read ndarray “a”
    # from “file”
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

- Blue variables are Object IDs
- Similar to futures



Task graph executed to compute `sum`

Example: In-app promotion



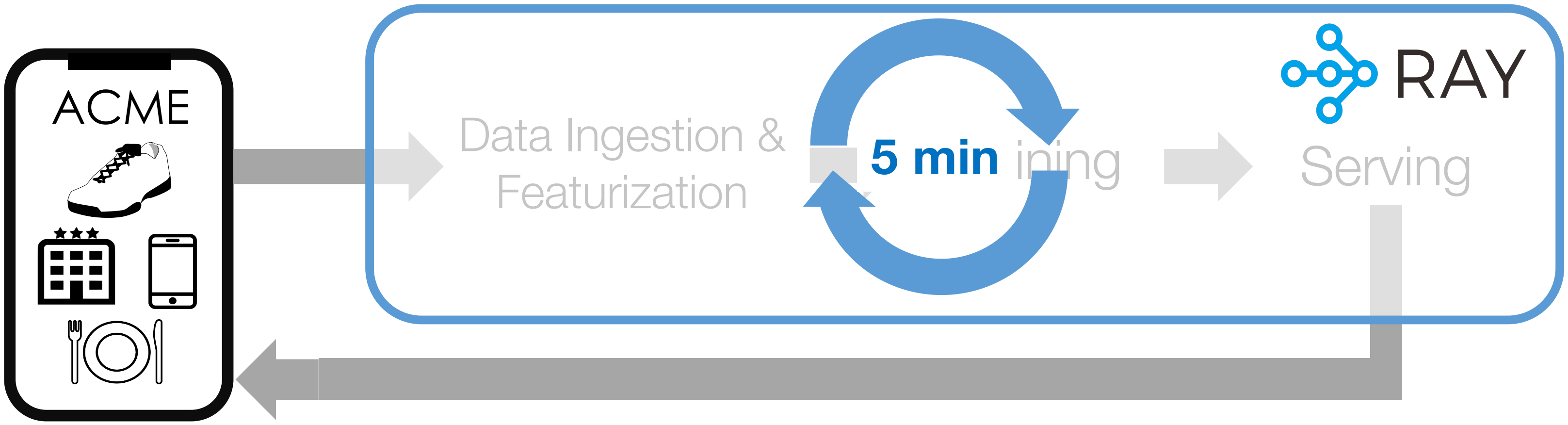
Model updated every **1 day**

+ 5% CTR

Model updated every **1 hour** (using state-of-the-art solution)

?

Ray: unified platform for distributed apps



Model updated every **1 day**

\+ **5%** CTR

Model updated every **1 hour** (using state-of-the-art solution)

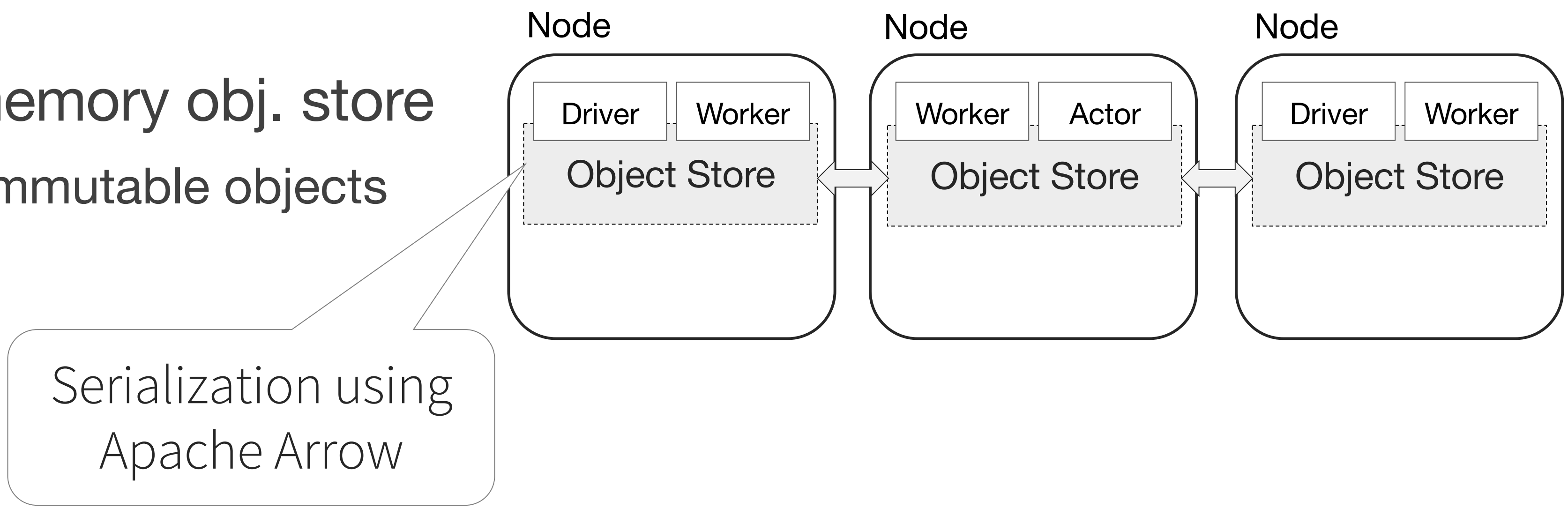
\+ **1%** CTR

Model updated every **5 min** using **Ray**

Ray Architecture

In-memory obj. store

- Immutable objects

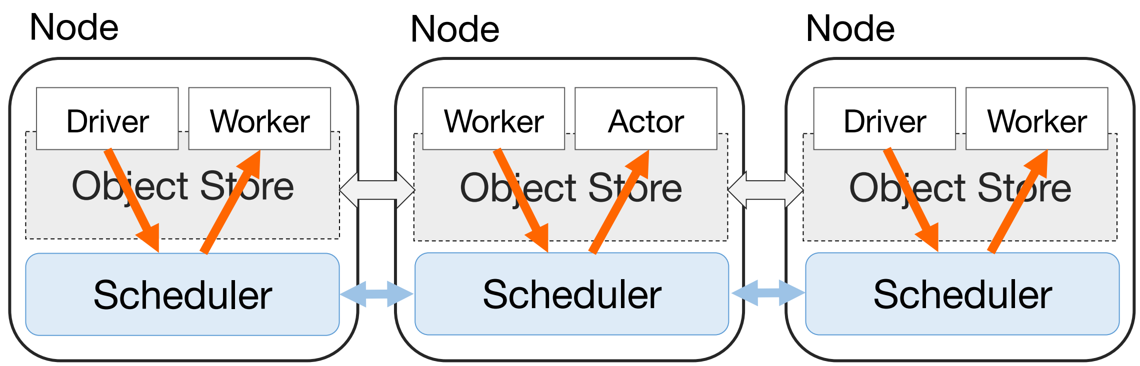


Ray Architecture

In-memory obj. store

- Immutable objects

Distributed scheduler

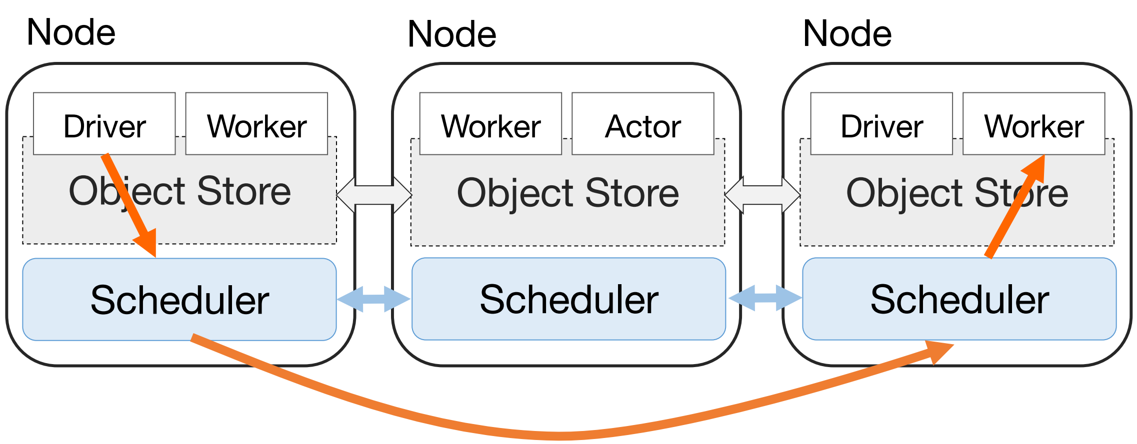


Ray Architecture

In-memory obj. store

- Immutable objects

Distributed scheduler



Ray Architecture

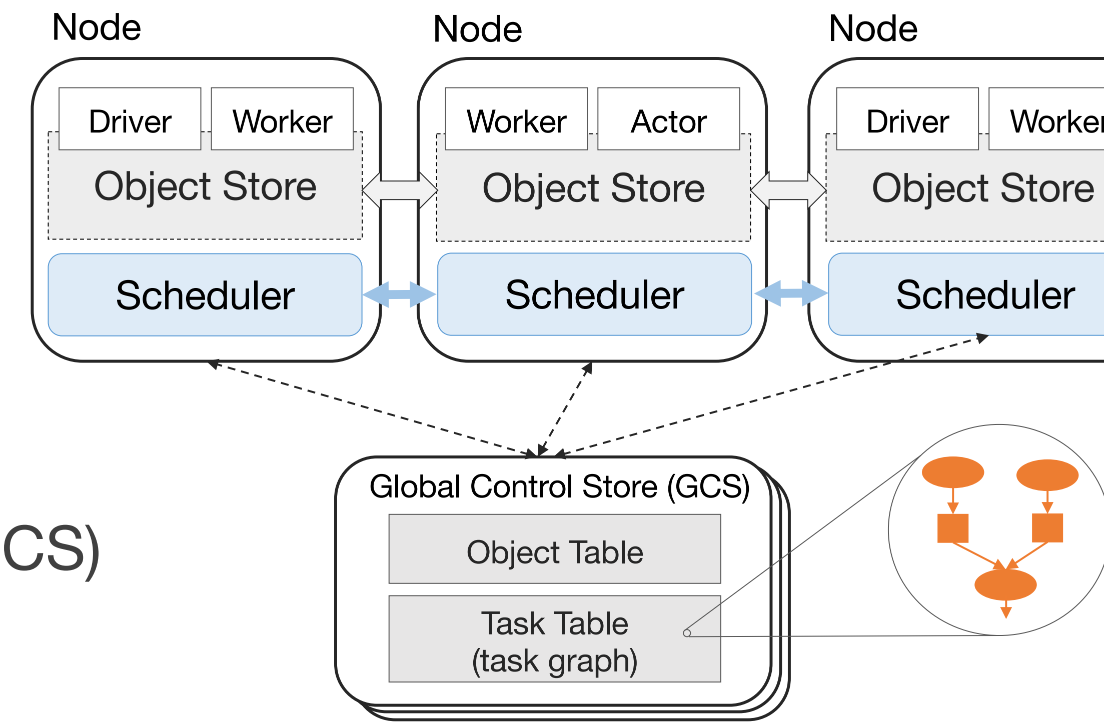
In-memory obj. store

- Immutable objects

Distributed scheduler

Central control store (GCS)

- Stateless components



Ray Architecture

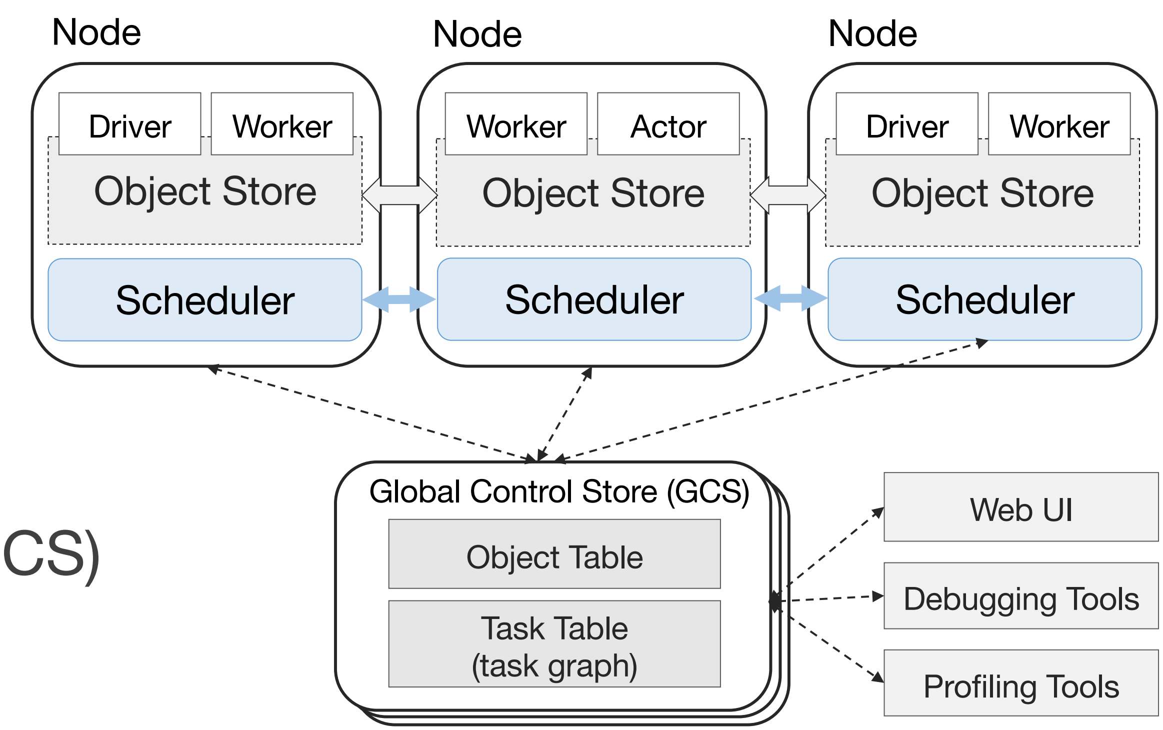
In-memory obj. store

- Immutable objects

Distributed scheduler

Central control store (GCS)

- Stateless components



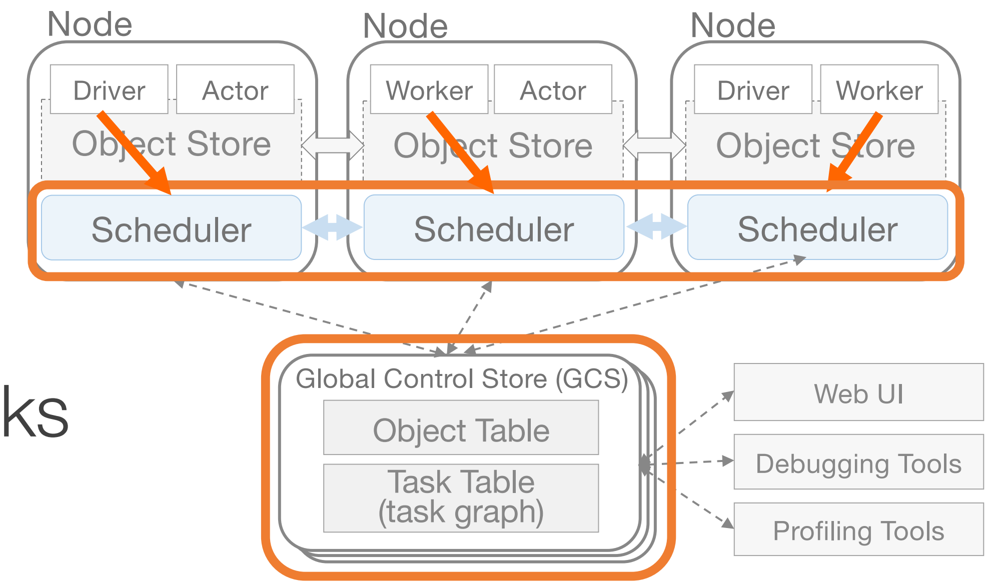
Scalability

Decentralized scheduler

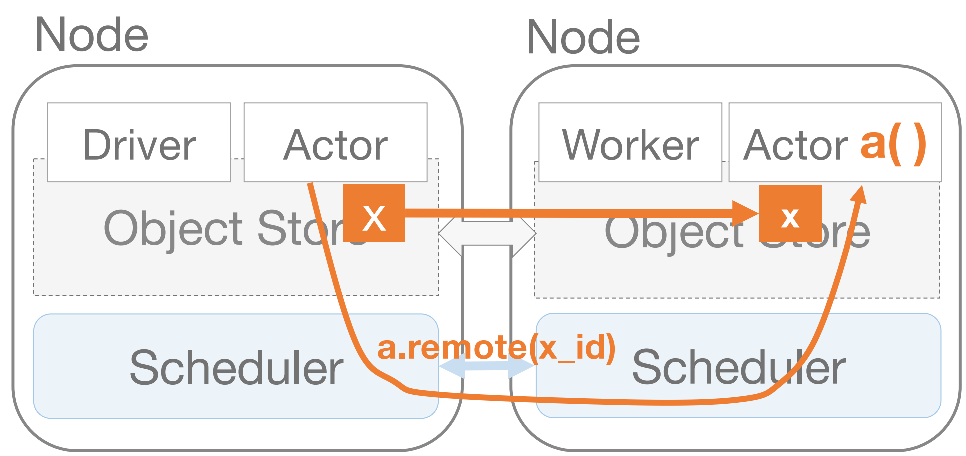
Sharded GCS

Any worker can submit tasks

- Driver not a bottleneck



Scalability



Decentralized scheduler

Sharded GCS

Any worker can submit tasks
- Driver not a bottleneck

Actors arguments sent by reference instead inline
- Avoid unnecessary copies
- Can optimize transfers (e.g., parallel transfers, multicast)

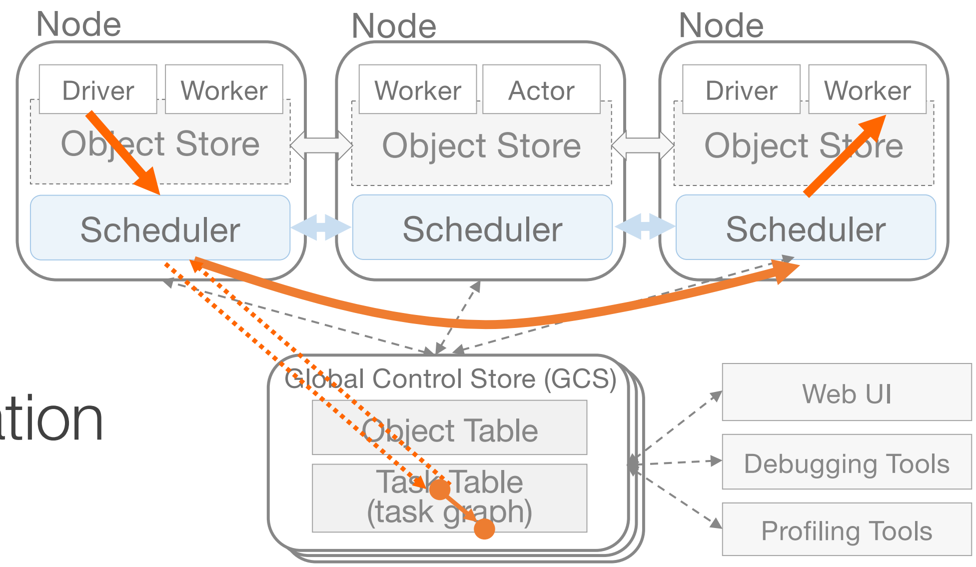
Fault Tolerance

Lineage based

- Replay computation to reconstruct lost objects

Classic solution

- Store lineage before invocation
- Insert delay on critical path

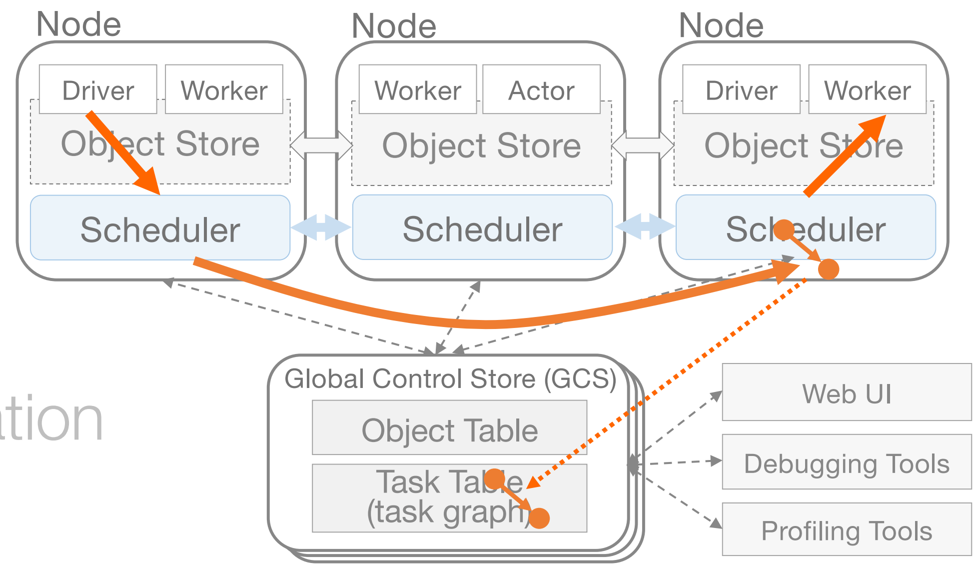


Fault Tolerance

Lineage based
- Replay computation to reconstruct lost objects

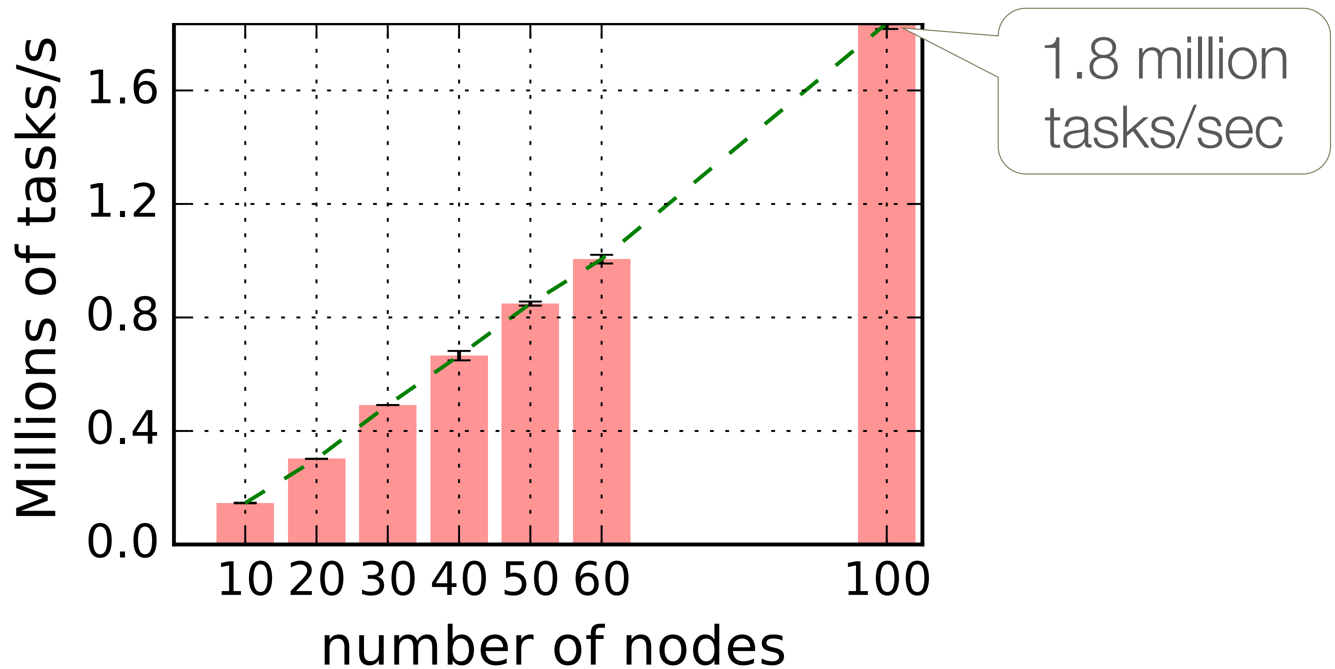
Classic solution
- Store lineage before invocation
- Insert delay on critical path

Lineage stash*
- Store lineage in task call and flush it



*"Lineage Stash: Fault Tolerance Off the Critical Path", Stephanie Wang et al, SOSP '19

Scalability & Performance



Latency of local task execution: ~300 us

Latency of remote task execution: ~1ms

Ray vs specialized systems

Match performance of specialized systems

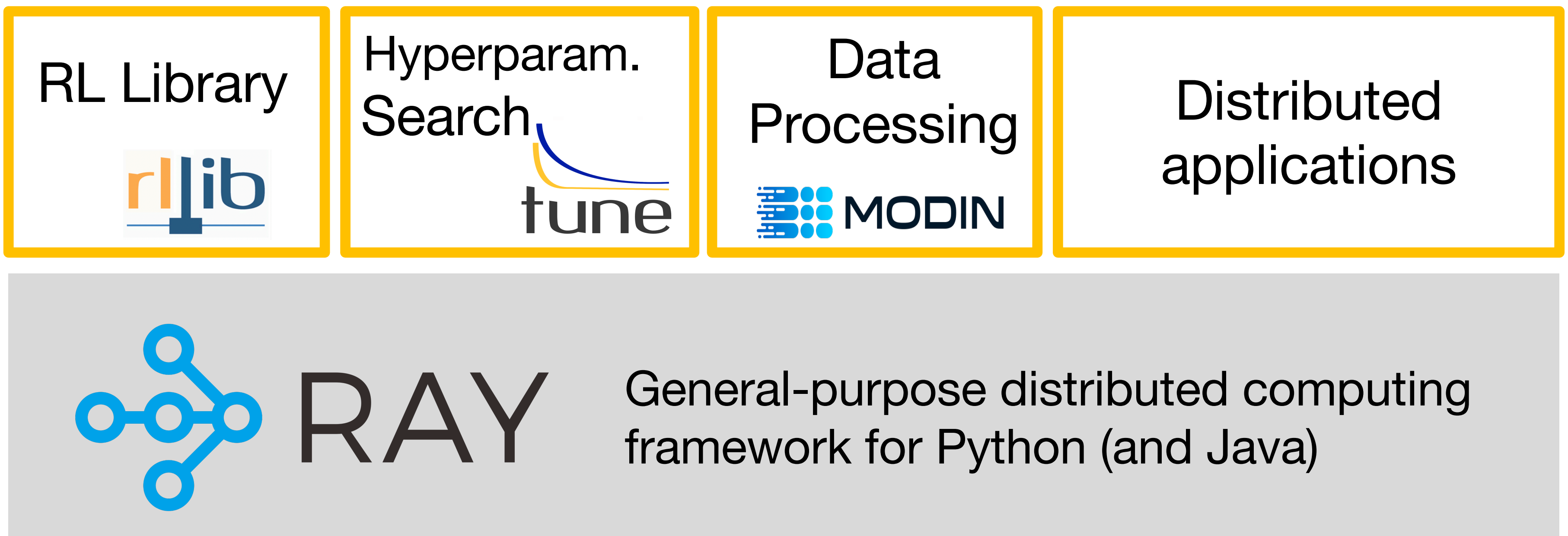
Ongoing work

Flexible scheduling policies

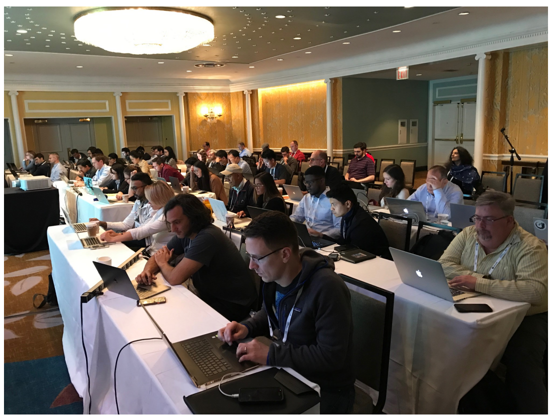
- Need to support *conflicting* policies, e.g., affinity, antiaffinity, locality, gang scheduling, …

Improved garbage collection for object store

- Currently LRU, but not ”good enough”
- Ideally, global reference counting


RL Library
rllib
Hyperparam.
Search
tune
Data
Processing
MODIN
Distributed
applications
RAY
General-purpose distributed computing
framework for Python (and Java)

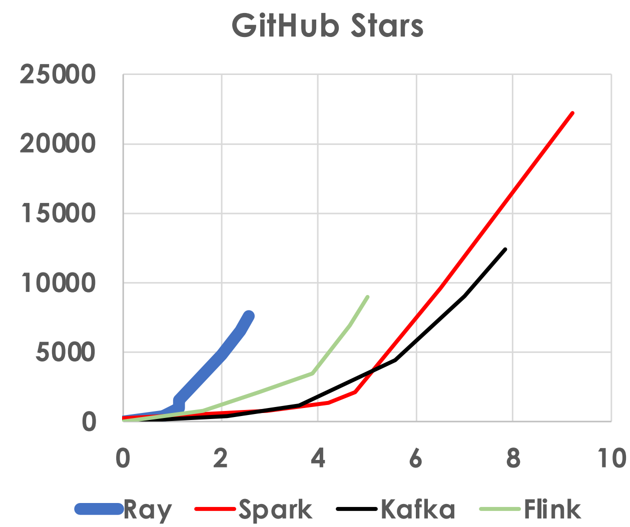
Growing adoption



200 contributors from
40+ companies

Sold out tutorials at O'Reilly AI

Included in AWS Sage Maker



Systems **ML**

- Ray

Systems **ML**

- Deep queries
- Neurocuts
- Autopandas

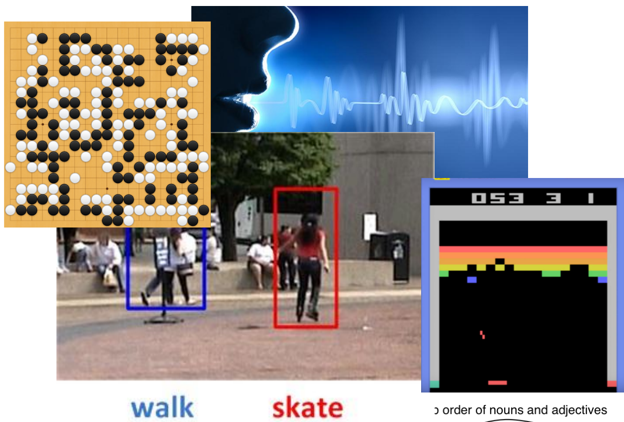
Systems **ML**

- Ray

Systems **ML**

- Deep queries
- Neurocuts
- Autopandas

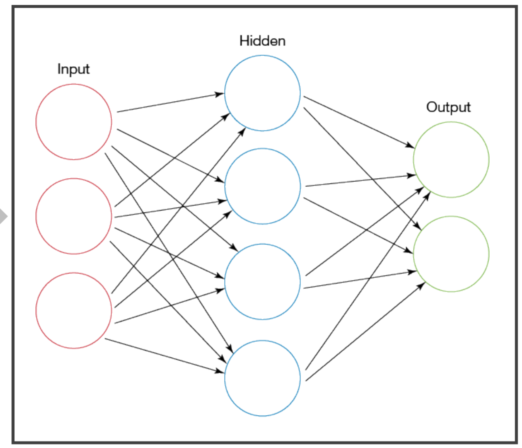
“Classic” DL/RL apps



order of nouns and adjectives

Quiero ir a la playa más bonita.

I want to go to the prettiest beach.



Speech recognition

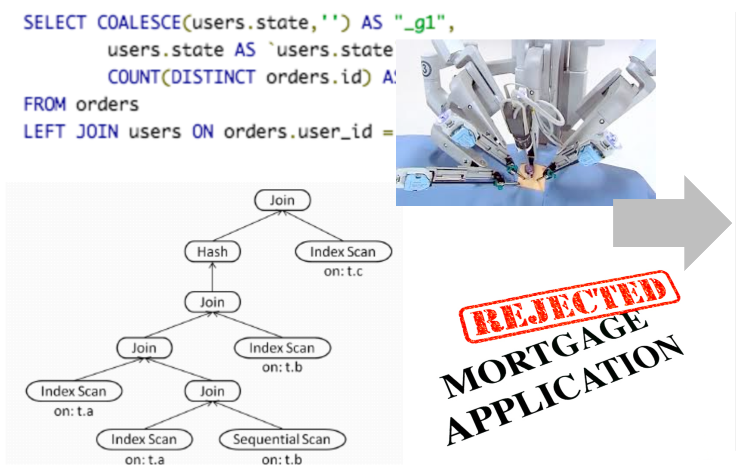
Video recognition

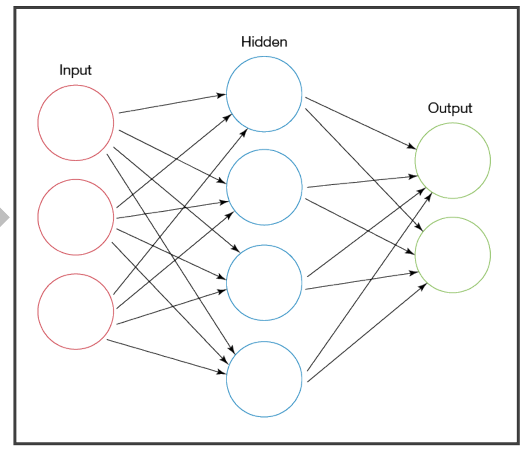
Language translation

…

Human tasks: 100% accuracy not expected

Systems problems





Program synthesis

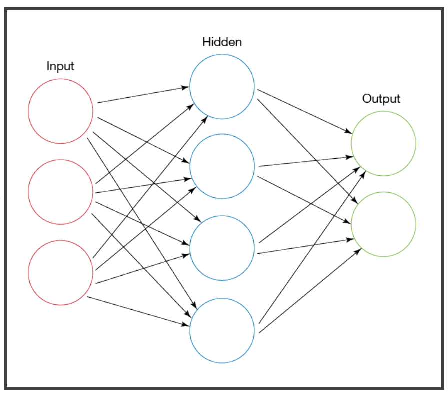
Mortgage decisions

Robotic surgery

...

Must ensure correctness and explainability!

Three approaches



Correct by construction — Build a solution that is provably correct

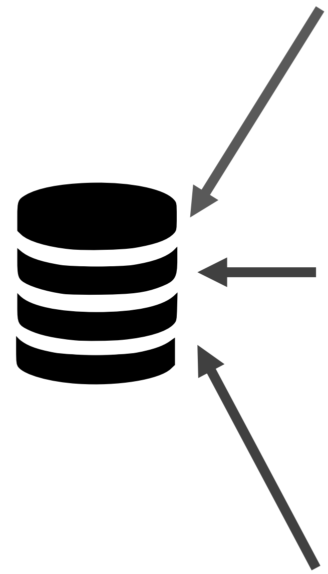
Iterative optimization — Sequence of transformations w/o impacting correctness

Generate & Verify — Generate solutions whose correctness can be verified

Database joins

Calculate tax owned by
“`Manager I`” employees

```
SELECT SUM(sal.salary*tax.rate)
FROM emp, sal, tax
WHERE emp.position = sal.position AND
      tax.country = sal.country AND
      emp.position = ‘Manager I’
```



emp_id	position	country
1	Manager II	USA
2	Engineer I	CAN
3	Engineer II	USA
	…	..

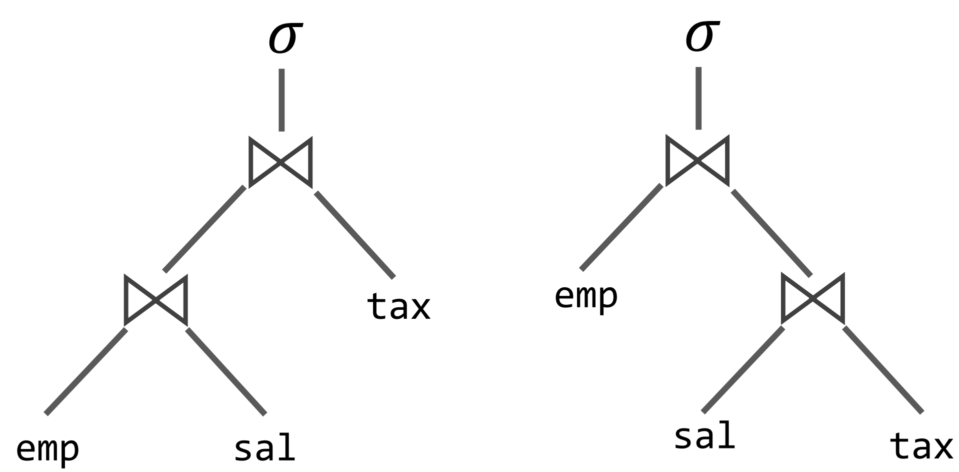
sal_id	position	salary
1	Manager I	120000.00
2	Manager II	150000.00
3	Engineer I	78000.00
4	Engineer II	91000.00

tax_id	country	rate
1	USA	0.32
2	CAN	0.45
3	CHN	0.17
4	…	…

Join Optimization

```
SELECT SUM(sal.salary*tax.rate)
FROM emp, sal, tax
WHERE emp.position = sal.position AND
      tax.country = sal.country AND
      emp.position = ‘Manager I’
```

In what order do you perform joins?



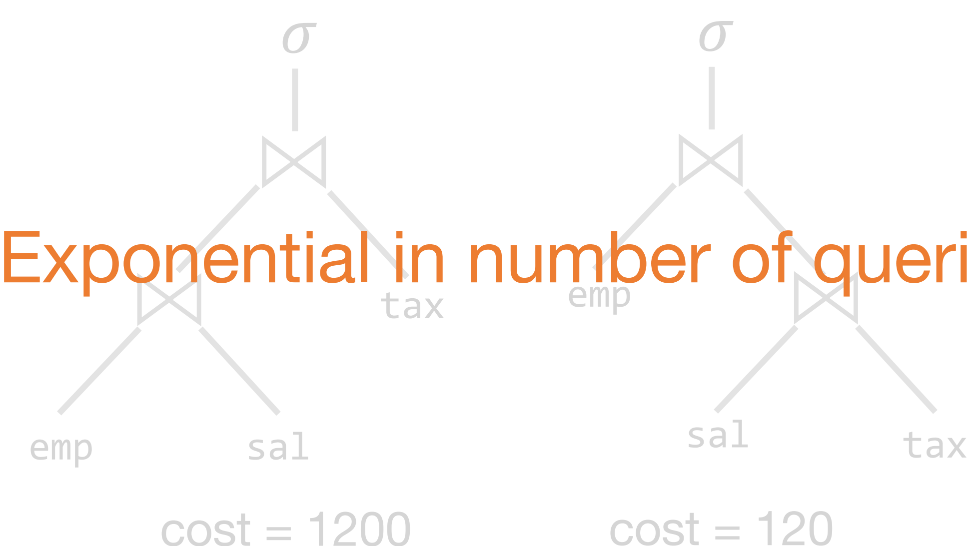
cost = 1200

cost = 120

Join Optimization

```
SELECT SUM(sal.salary*tax.rate)
FROM emp, sal, tax
WHERE emp.position = sal.position AND
      tax.country = sal.country AND
      emp.position = 'Manager I'
```

In what order do you perform joins?



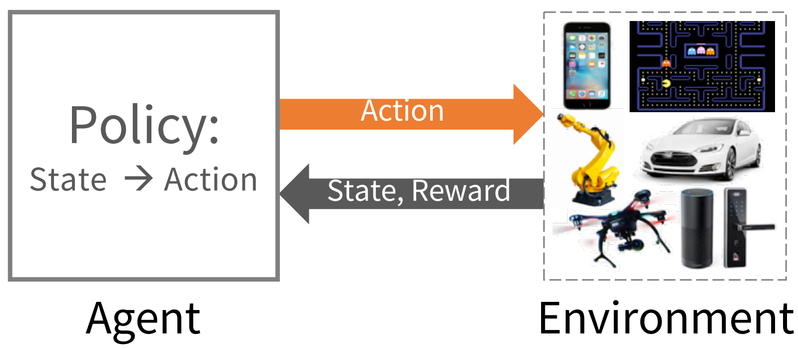
Exponential in number of queries!

40 years of heuristics!

Left-deep: maximize index usage
Right-deep: maximize hash table re-use
Zig-zag: union of LD and RD
Greedy: exploit linear cost models
IK-KBZ: exploit Star Schemas
GEQO: genetic algorithms in Postgres

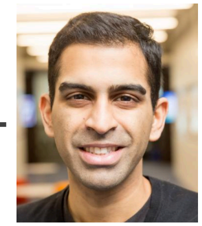
Can ML replace programmed heuristics
with efficient and data-driven strategies?

Reinforcement Learning (RL)



Agent continually learning by interacting with env.
Compute **policy** (i.e., **state → action**) to maximize **reward**

Deep Query*: Join plans with Deep RL



Why reinforcement learning (RL)?

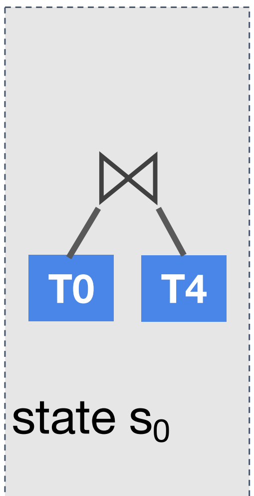
- Natural formulation
  - **State**: set of tables joined so far
  - **Action**: table to join next
  - **Reward**: negative of estimated cost
- Use Deep Q-Learning

*"Learning to Optimize Join Queries With Deep Reinforcement Learning", Sanjay Krishnan et al (https://arxiv.org/abs/1808.03196)

Deep Query: Build join plans with Deep RL

Why reinforcement learning (RL)?

- Natural formulation

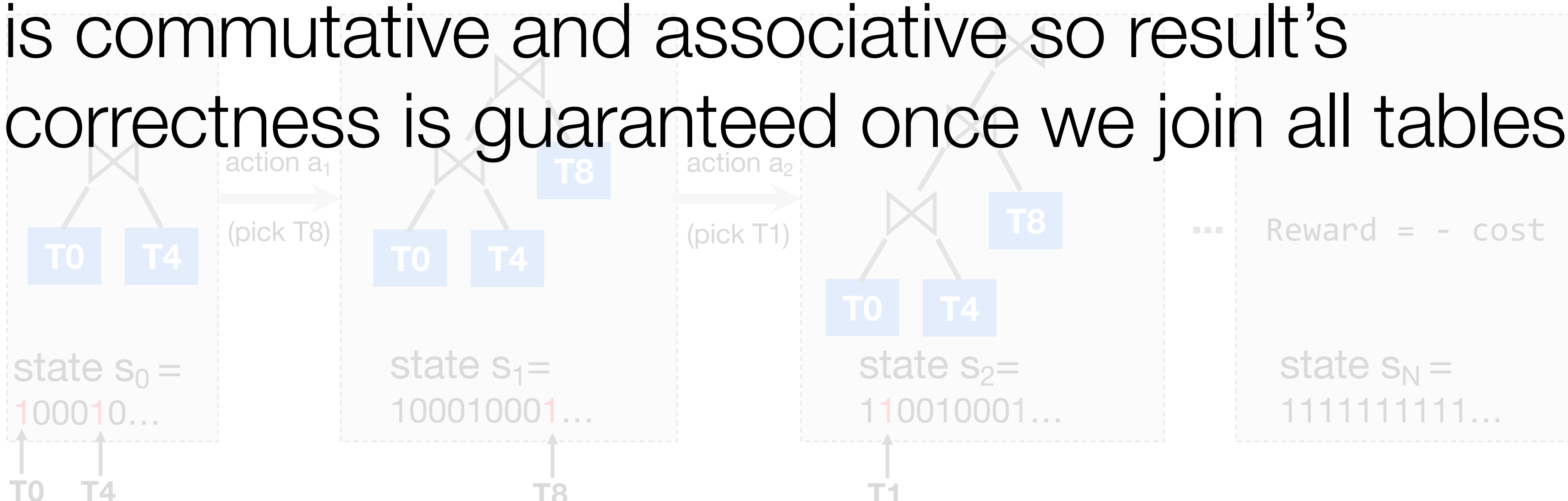


Deep Query: Build join plans with Deep RL

Why reinforcement learning (RL)?

- Natural formulation

Correct by construction: the ”join” operation is commutative and associative so result’s correctness is guaranteed once we join all tables



Deep Query: Properties

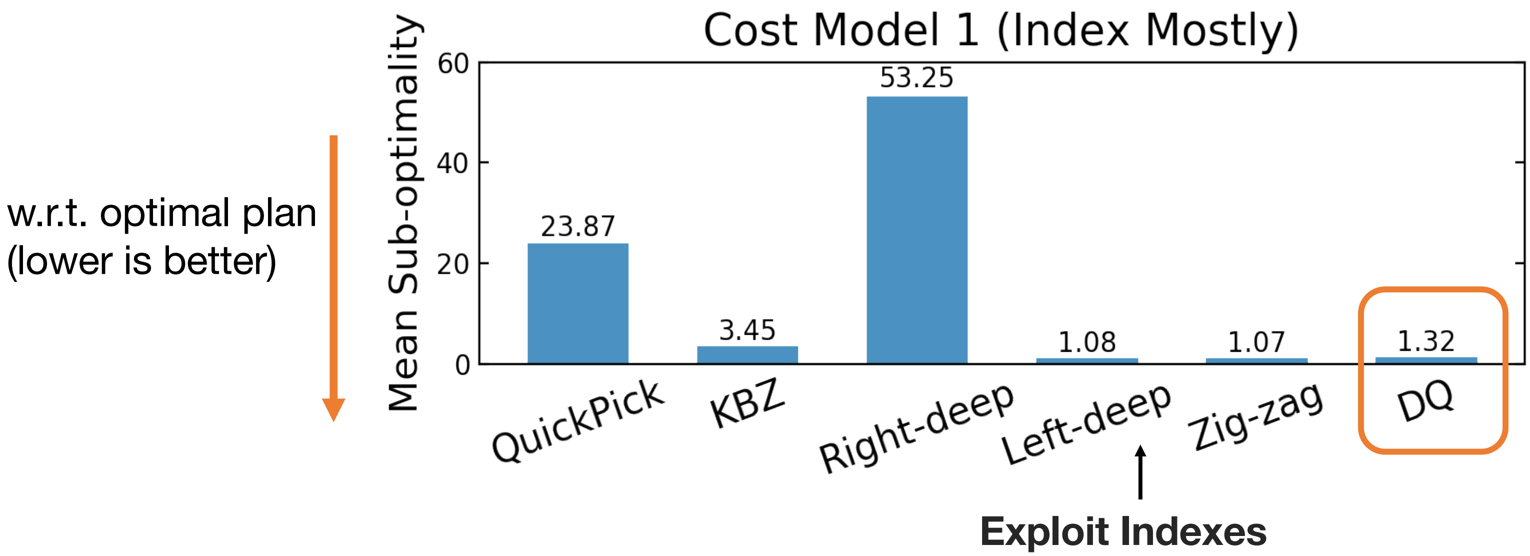
Generalize to unseen queries

Adapt to workload and hardware characteristics

Efficient planning (order of magnitude faster)

113 queries, IMDb dataset, 21 tables
4–17 way joins (avg ~8 relations per query)
Cost models:

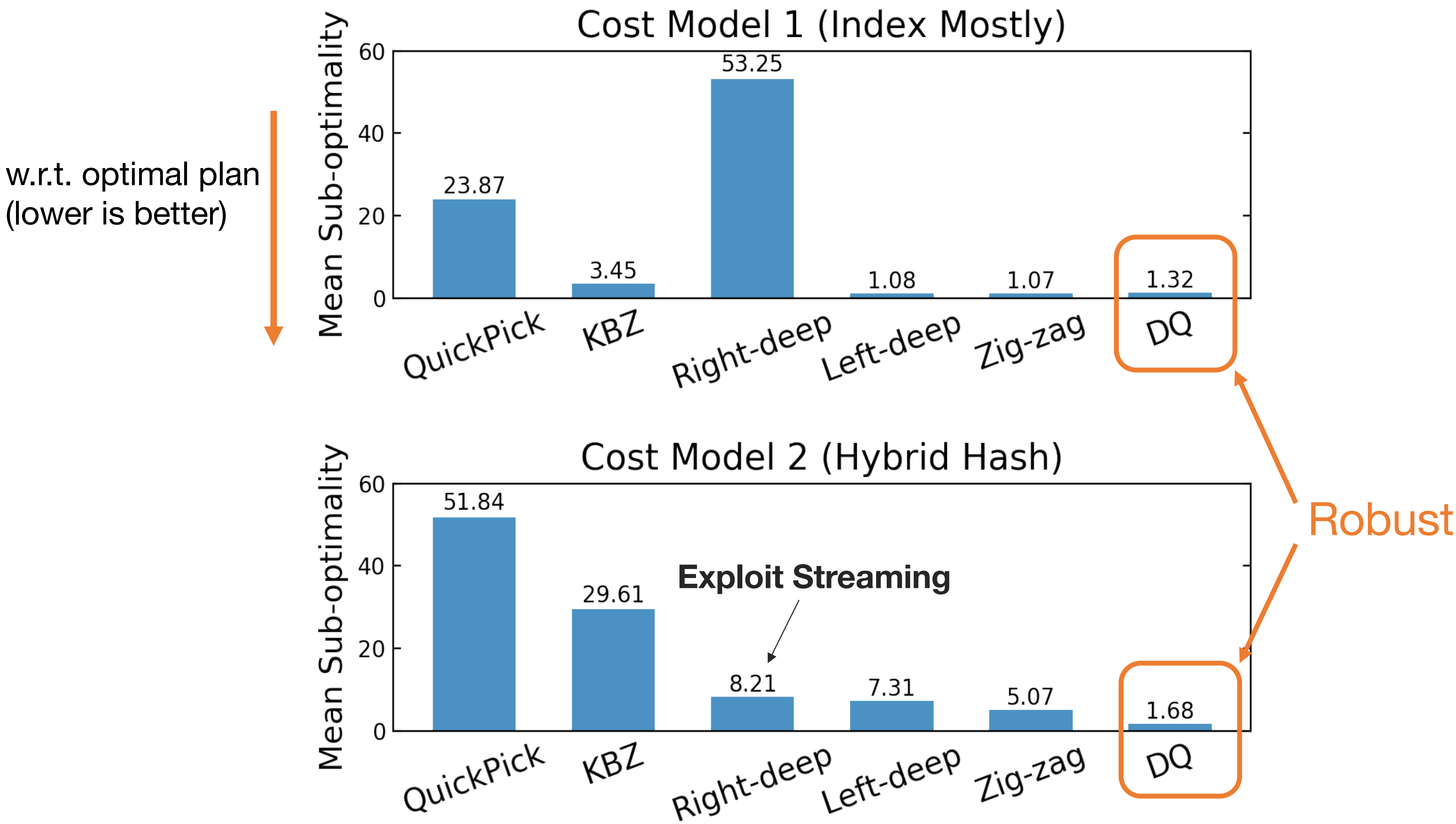
- Model 1: Index Mostly
- Model 2: Hybrid Hash



113 queries, IMDb dataset, 21 tables
4–17 way joins (avg ~8 relations per query)
Cost models:

- Model 1: Index Mostly
- Model 2: Hybrid Hash

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems


w.r.t. optimal plan
(lower is better)
Cost Model 1 (Index Mostly)
Mean Sub-optimality
60
40
20
0
23.87
3.45
53.25
1.08
1.07
1.32
QuickPick
KBZ
Right-deep
Left-deep
Zig-zag
DQ
Robust
Cost Model 2 (Hybrid Hash)
Mean Sub-optimality
60
40
20
0
51.84
29.61
Exploit Streaming
8.21
7.31
5.07
1.68
QuickPick
KBZ
Right-deep
Left-deep
Zig-zag
DQ

Systems **ML**

- Ray

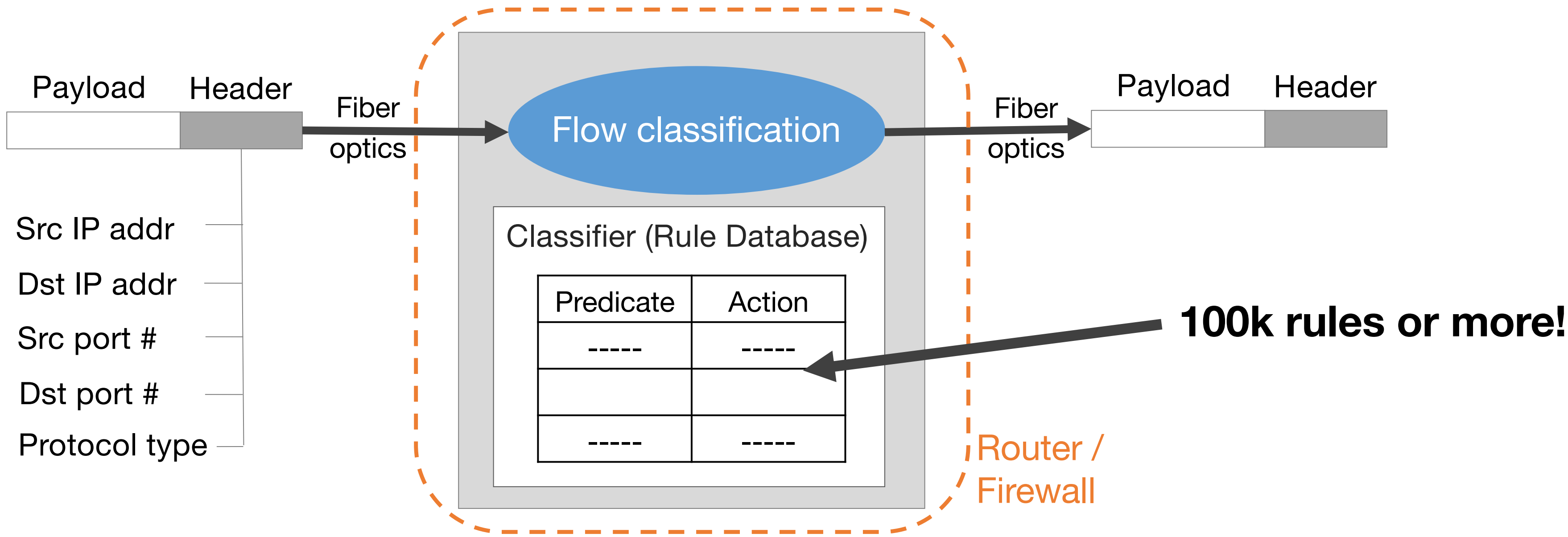
Systems **ML**

- Deep queries
- **Neurocuts**
- Autopandas

Packet Classification

Fundamental problem in networking

- Building block for access control, QoS, defense against attacks

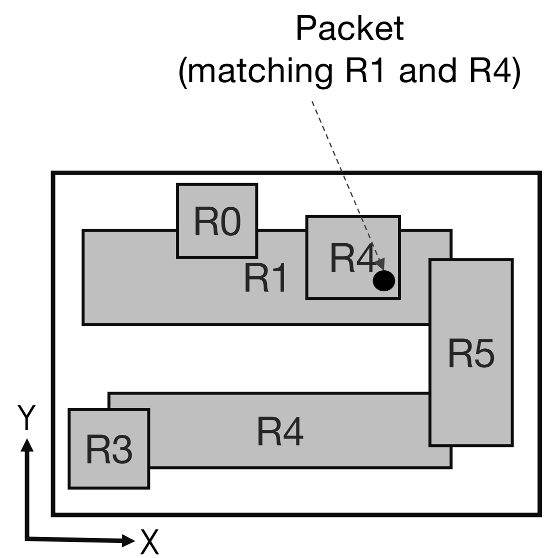


The problem

Similar to point-location in a hypercube

Hard time-space tradeoff:

- O(log N) time and O(N^d) space
- O($\log^{d-1} N$) time and O(N) space
- N: # of rules; d: # of attributes
  - In our case: N ~= 100K, d = 5



But harder: rules overlap and have priorities

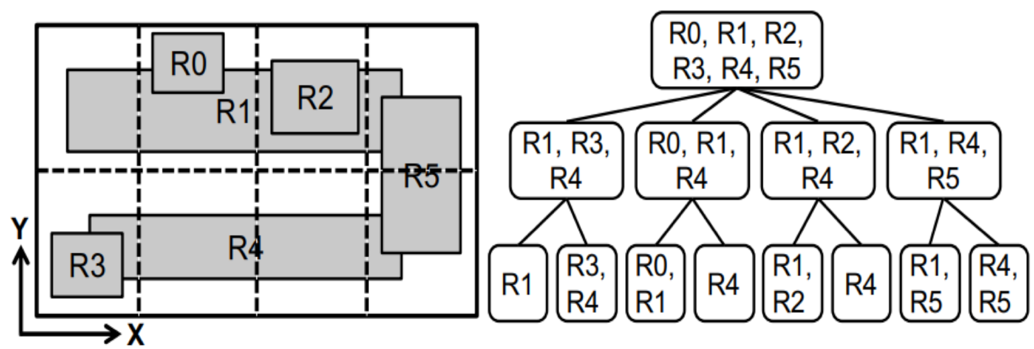
20+ years of work

Hardware

- Expensive and power hungry → prohibitive for large classifiers

Software

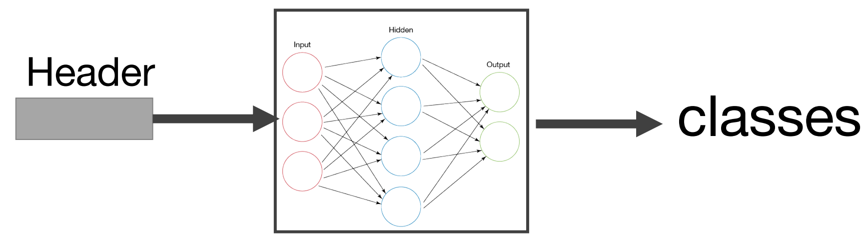
- Build a multi-dimensional "decision tree" – really a k-d index
- HiCuts ('99), HyperCuts ('03), EffiCuts ('10), CutSplit ('15)
  - All rely on hand-tuned heuristics, which are brittle and not optimal



(a) Packet classifier. (b) Decision tree.

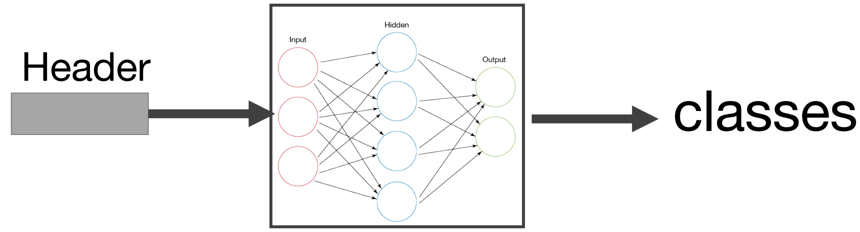
Two approaches

1. End-to-end solution

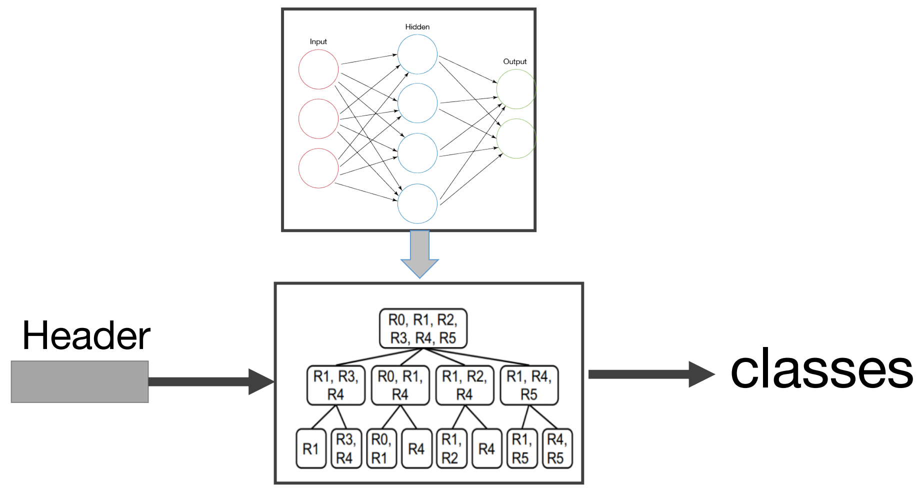


Two approaches

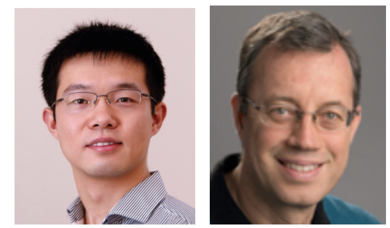
1. End-to-end solution



2. Build classification tree



NeuroCuts*: Building decision trees with Deep RL

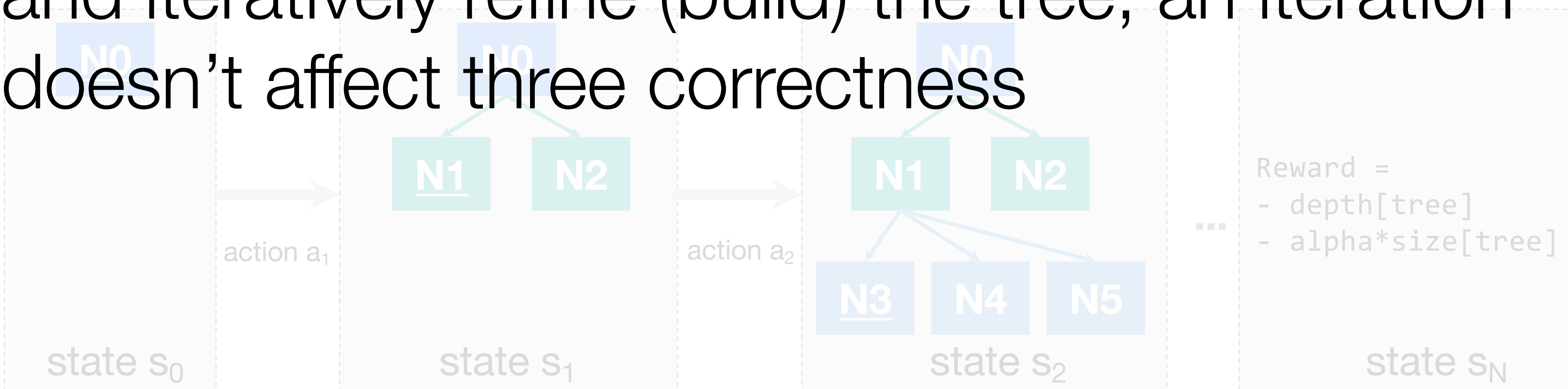


*"Neural Packet Classification", Eric Liang et al, SIGOMM 2019

NeuroCuts: Building decision trees with Deep RL

Why reinforcement learning (RL)?

- **Simulation is reality:** build a tree in simulation → deploy to
- Naive MDP (Markov Decision Process) formulation



Iterative optimization: start from a single-node and iteratively refine (build) the tree; an iteration doesn't affect three correctness

Results

Classification time: median 18% faster than state-of-the-art

3x better either in space or time than any previous solution

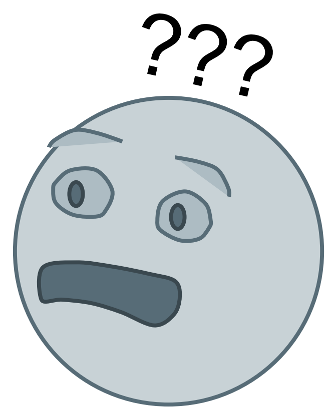
Systems **ML**

- Ray

Systems **ML**

- Deep queries
- Neurocuts
- **Autopandas**

API Explosion!



How to cope? *StackOverflow*

How do I turn this:

	weight	
	kg	lbs
cat	1	2
dog	2	4

into this:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4

in pandas?

Problems with *StackOverflow*

How do I turn this:

	weight	
	kg	lbs
cat	1	2
dog	2	4

into this:

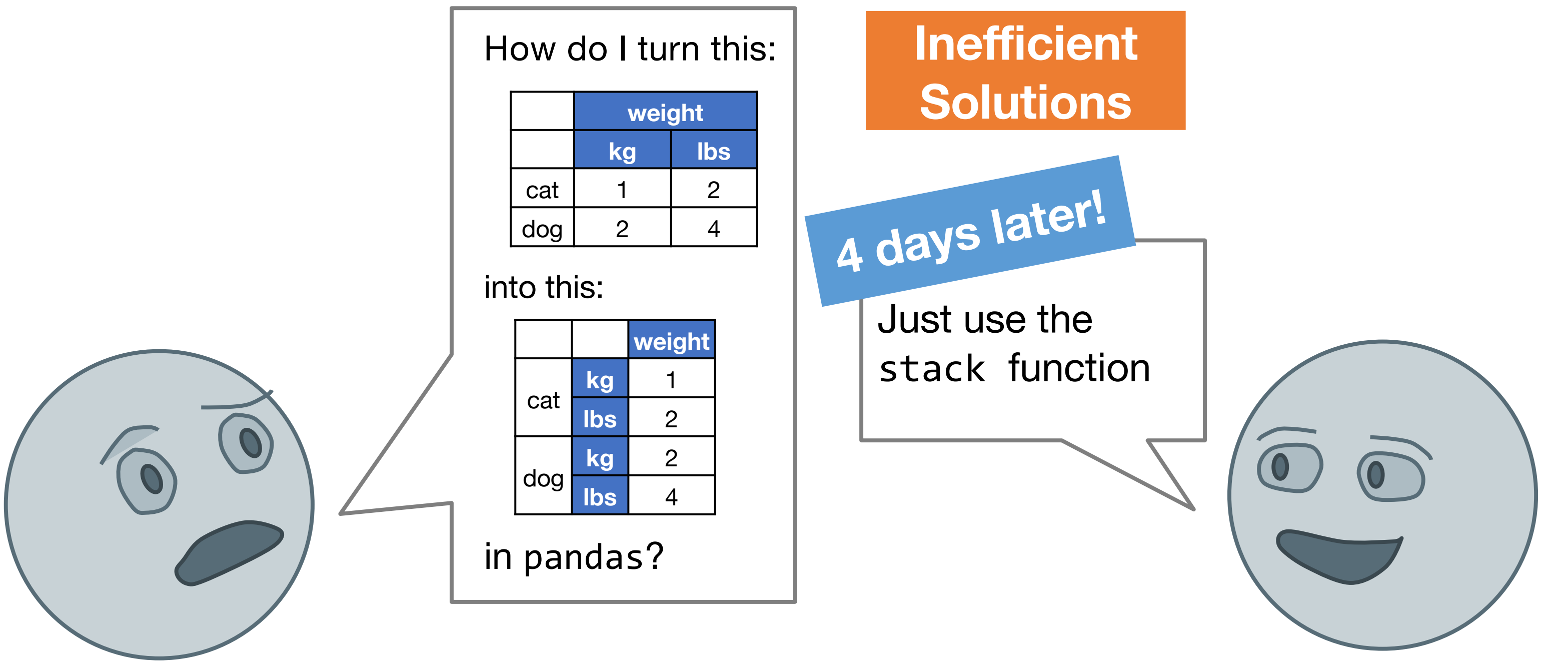
		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4

in pandas?

Inefficient Solutions

Well, you need to start by building the index `pd.multiIndex(...`

Problems with *StackOverflow*



Goal: *StackOverflow* for APIs via Program Synthesis

How do I turn this:

	weight	
	kg	lbs
cat	1	2
dog	2	4

into this:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4

in pandas?

```
output = input.stack(
            level=[1],
            dropna=True
         )
```

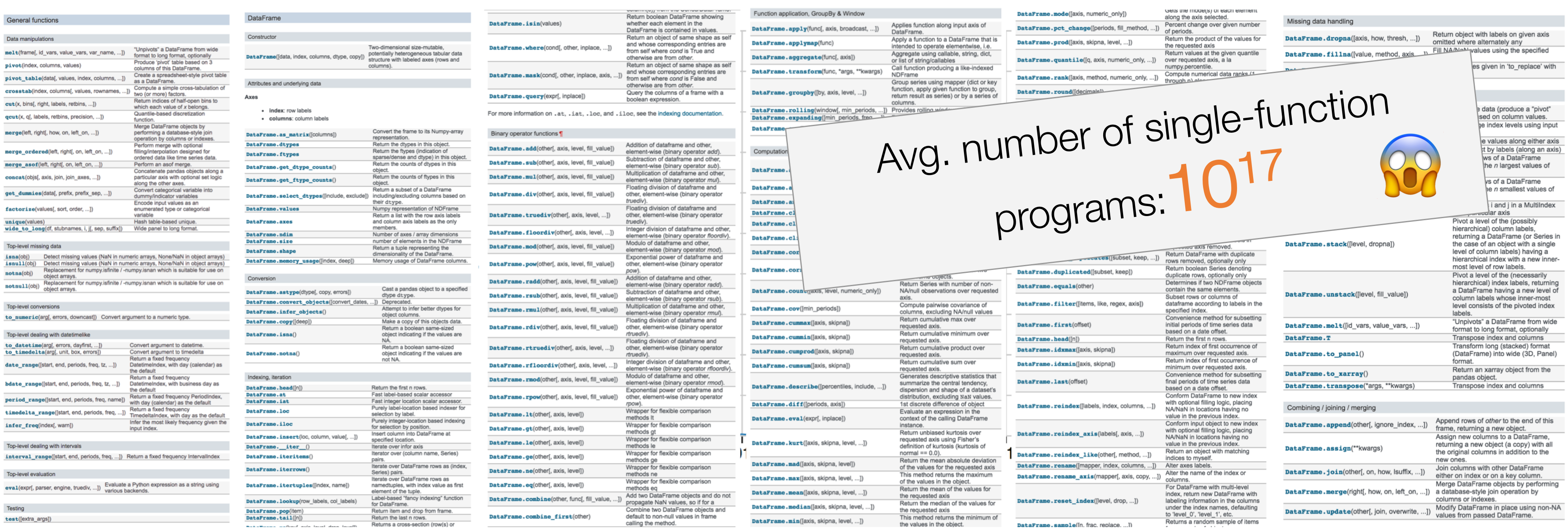
AutoPandas

Target API: *pandas* (DataFrame transformations)

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Premier library for data scientists



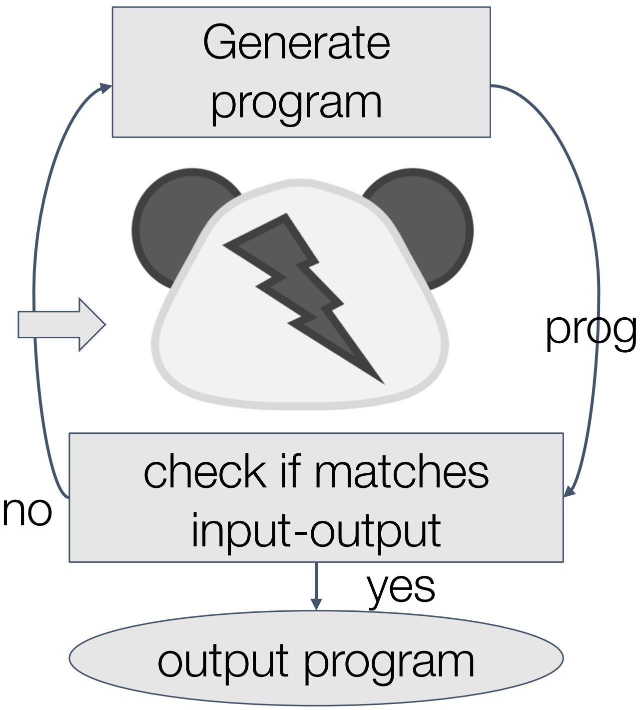
Autopandas*

input:

	weight	
	kg	lbs
cat	1	2
dog	2	4

output:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4



"AutoPandas: Neural-Backed Generators for Program Synthesis", Rohan Bavishi et al, OOPSLA 2019

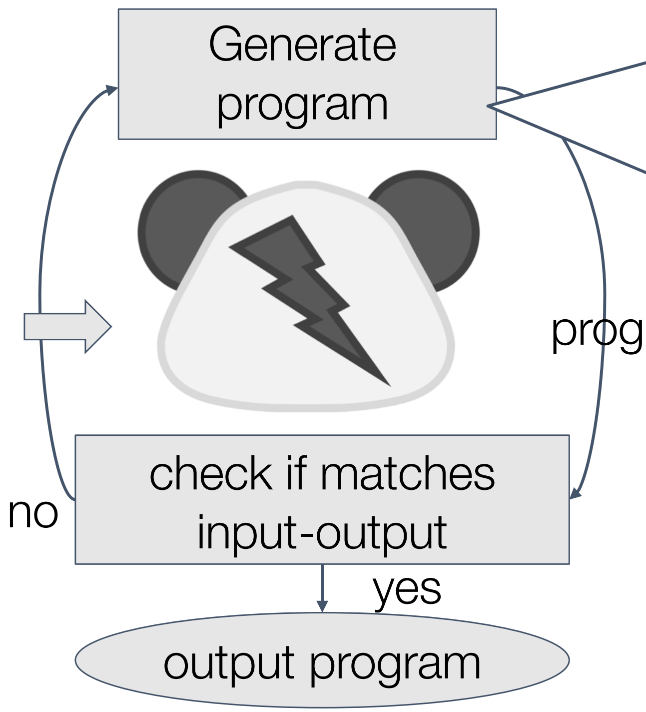
Predict program

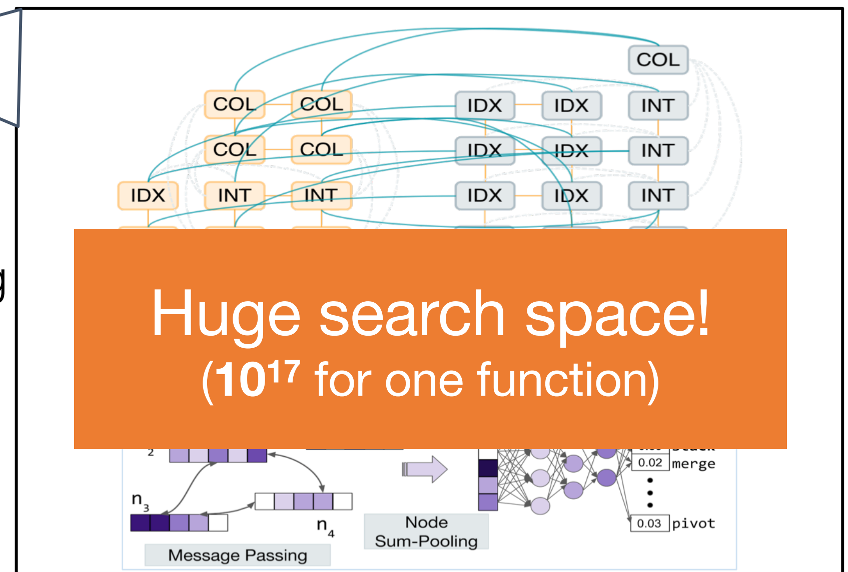
input:

	weight	
	kg	lbs
cat	1	2
dog	2	4

output:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4

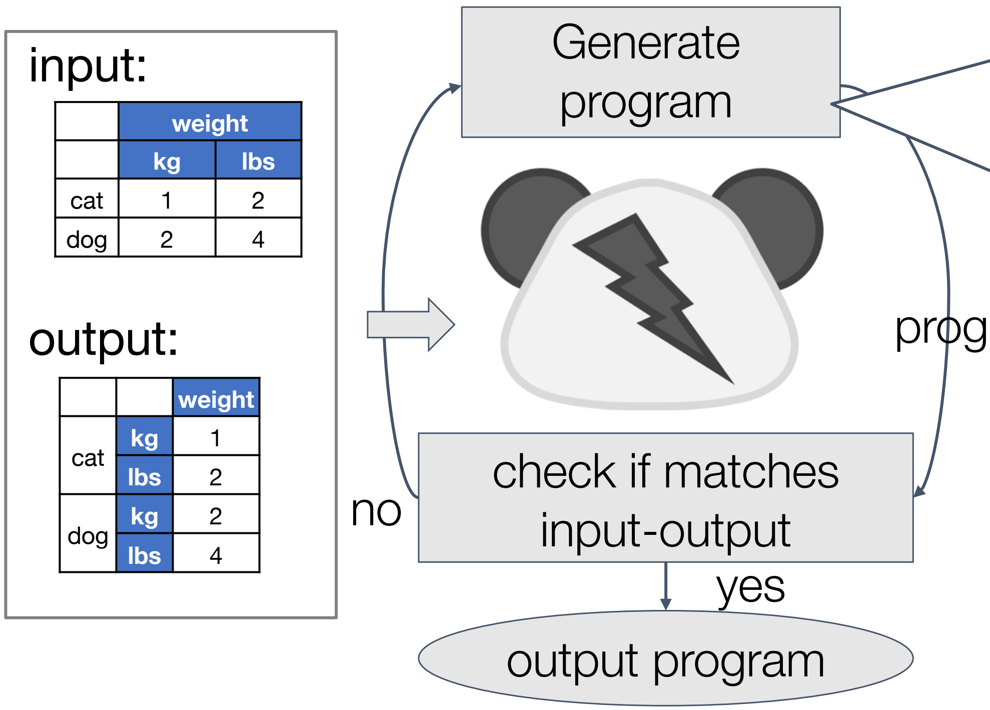




Graph Neural Networks*

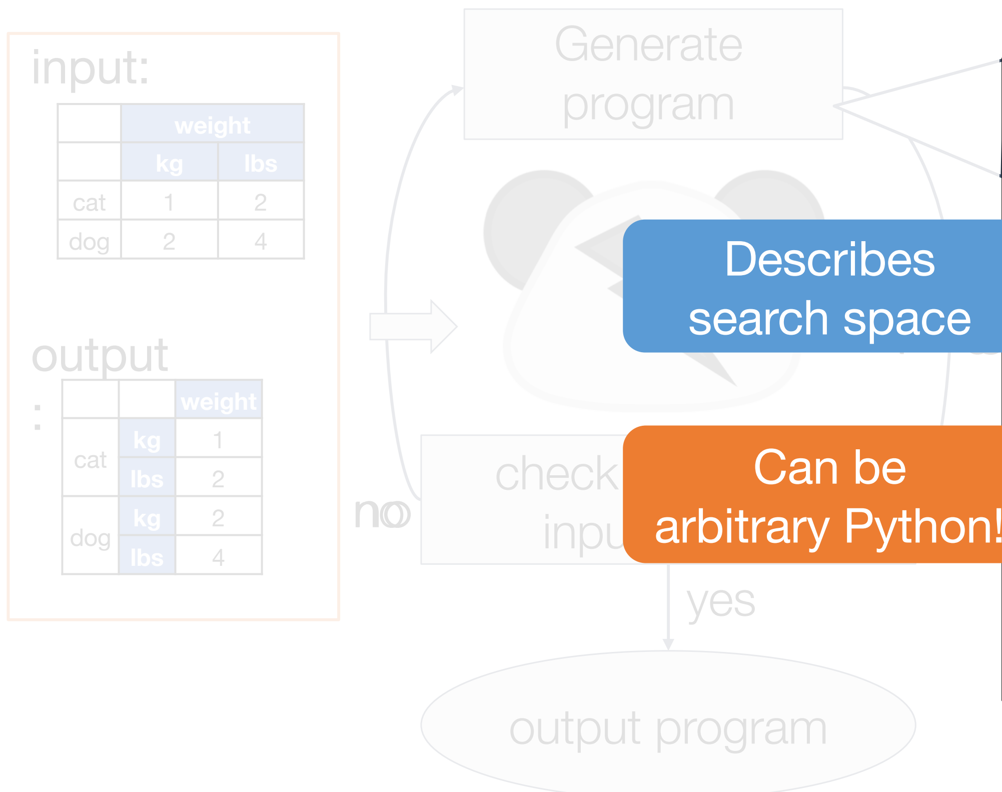
*M. Allamanis, M. Brockschmidt, and M. Khademi, Learning to represent programs with graphs, ICLR 2018

Neural-backed program generators




```
def generate_program(input, output):
  fn_seq = OrderedSequence(functions)
  for fn in fn_seq:
    if fn == pivot:
      arg_col = Select(input.cols)
      arg_idx = Select(input.cols
                       - {arg_col})
      fn.add_args(arg_col, arg_idx)
     elif fn == drop:
       arg_ax = Select({0,1})
       arg_lbl = Select(input.rows) if arg_ax
                     else Select(input.cols)
       fn.add_args(arg_ax, arg_lbl)
     elif ... :
       # <omitted code…>
  return fn_seq
```

Neural-backed program generators



Describes search space

Can be arbitrary Python!

```
def generate_program(input, output):
  fn_seq = OrderedSequence(functions)
  for fn in fn_seq:
    if fn == pivot:
      arg_col = Select(input.cols)
      arg_idx = Select(input.cols
                       - {arg_col})
      fn.add_args(arg_col, arg_idx)
    elif fn == drop:
      arg_ax = Select({0,1})
      arg_lbl = Select(input.rows) if arg_ax
                else Select(input.cols)
      fn.add_args(arg_ax, arg_lbl)
    elif ... :
      # <omitted code…>
  return fn_seq
```

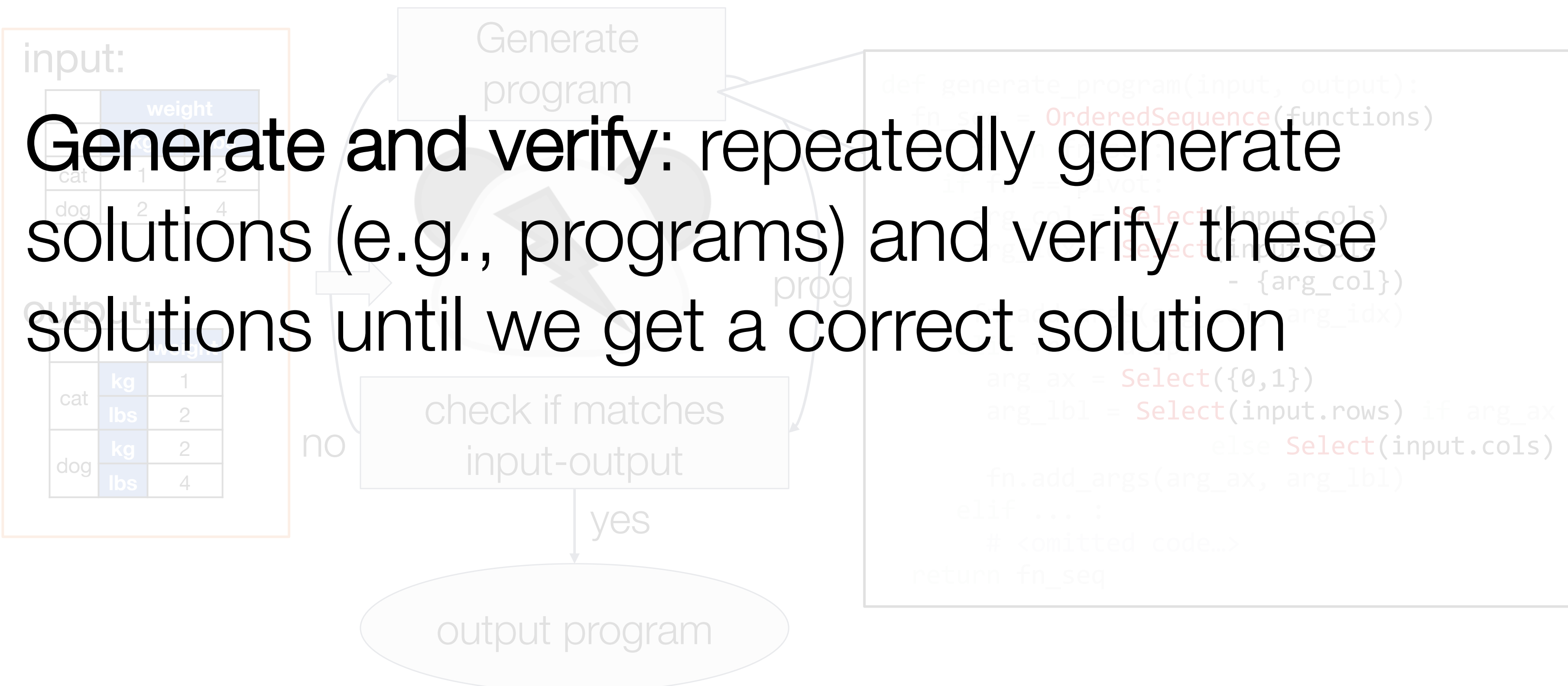
Neural-backed program generators


```
def generate_program(input, output):
  fn_seq = OrderedSequence(functions)
  for fn in fn_seq:
    if fn == pivot:
      arg_col = Select(input.cols)
      arg_idx = Select(input.cols
                  - {arg_col})
      fn.add_args(arg_col, arg_idx)
    elif fn == drop:
      arg_ax = Select({0,1})
      arg_lbl = Select(input.rows) if arg_ax
                  else Select(input.cols)
      fn.add_args(arg_ax, arg_lbl)
    elif ... :
      # <omitted code >
```

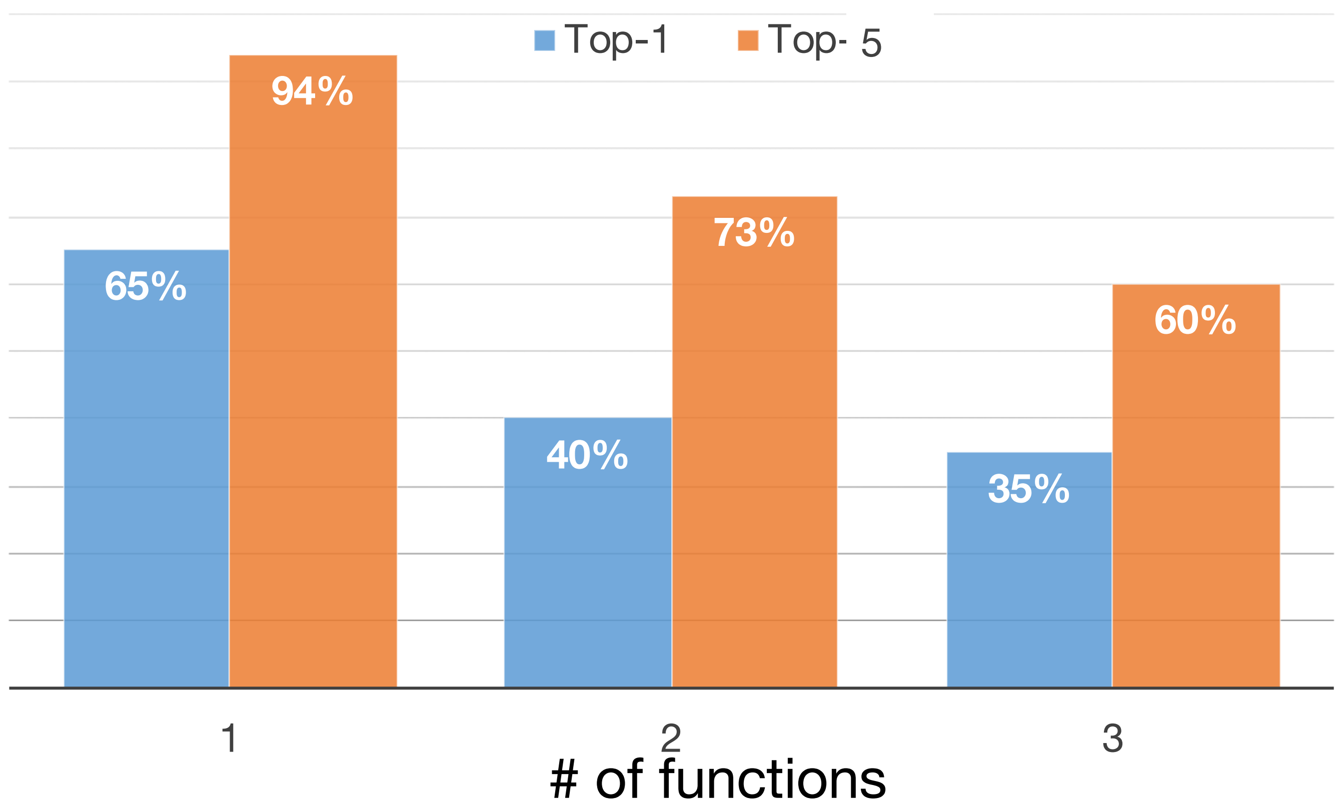
For one function reduce search space by 10^{12}

(from 10^{17} to 10^{5})

Neural-backed program generators



Generate and verify: repeatedly generate solutions (e.g., programs) and verify these solutions until we get a correct solution



- 70 benchmarks from StackOverflow, “Python for Data Analysis”, “Data School” videos
- ~95% of code snippets in StackOverflow have length <= 3 functions

Summary

Systems ML

ML needs systems: scale algorithms, easy to program

Systems need ML: improve state-of-the-art heuristics

Putting the two together → explosive growth

Systems → better ML → better Systems → …