Systems and ML at RISELab

Ion Stoica

UC Berkeley, Director of RISELab

July 13, 2020

riselab
UC Berkeley

Studies the design of

real-time,

intelligent,

secure, and

explainable

algorithms and systems

riselab
UC Berkeley

Studies the design of

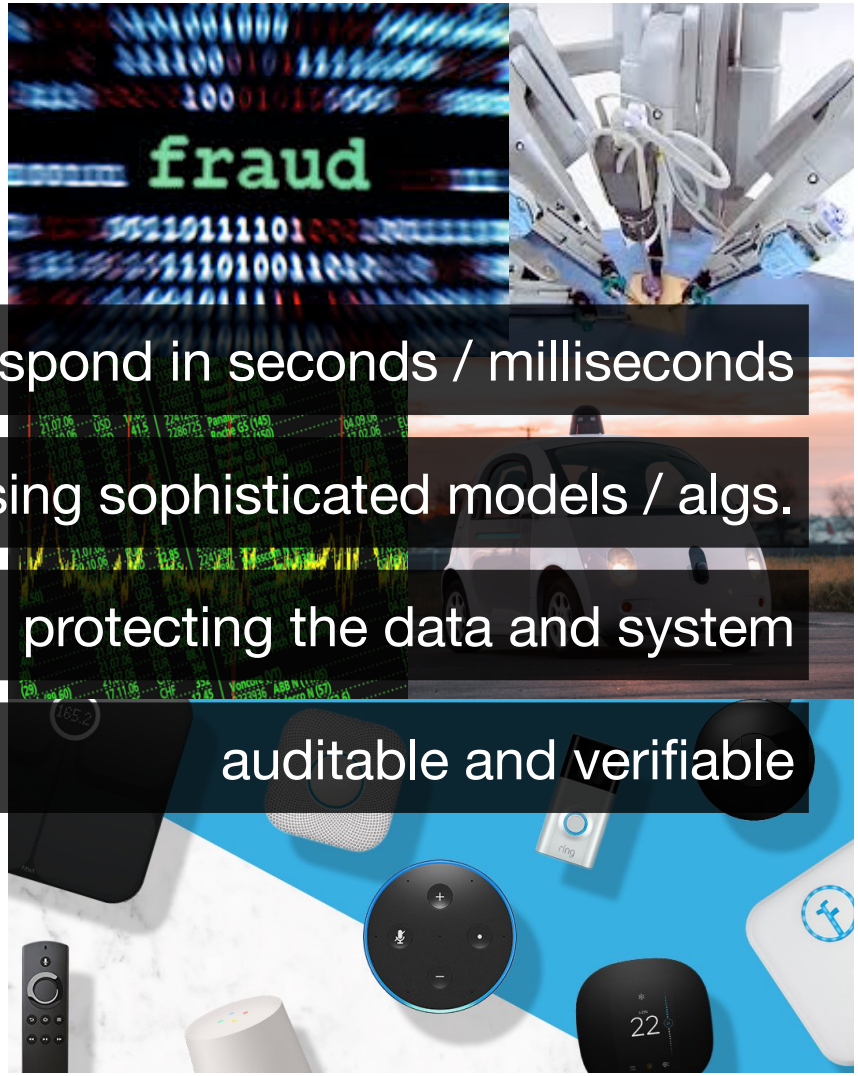
real-time,

intelligent,

secure, and

explainable

algorithms and systems



riselab
UC Berkeley

Studies the design of

real-time,

intelligent,

secure, and

explainable

algorithms and systems

Interdisciplinary Lab

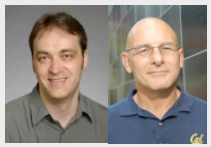
AI

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 CapitalOne ERICSSON

facebook Google intel Microsoft

NVIDIA Scotiabank splunk> vmware


SystemsML

- Ray

- Neurocuts
- Autopandas

- Ray

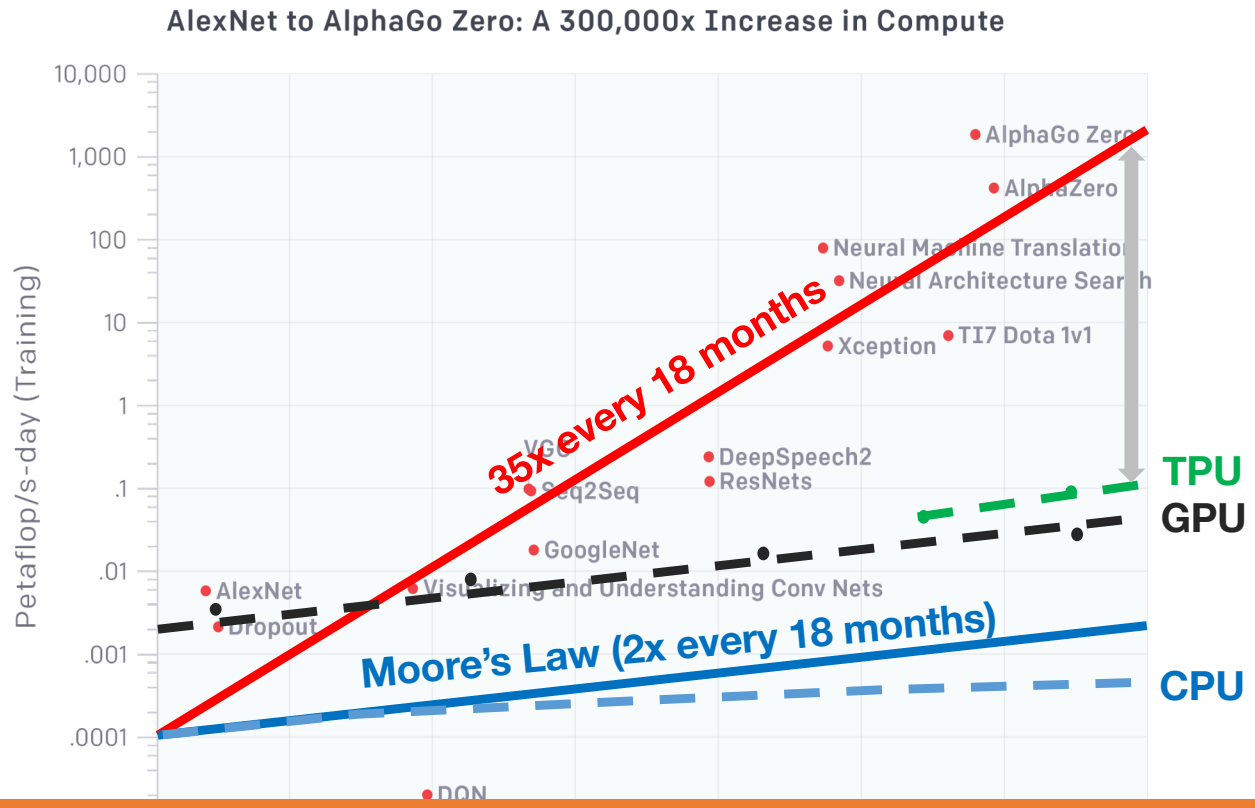
- Neurocuts
- Autopandas

Trends

Apps becoming **distributed**

Apps becoming more **complex**

Apps becoming **distributed**



No choice but to distribute apps

Apps becoming more **complex**

Virtually all apps will become AI centric

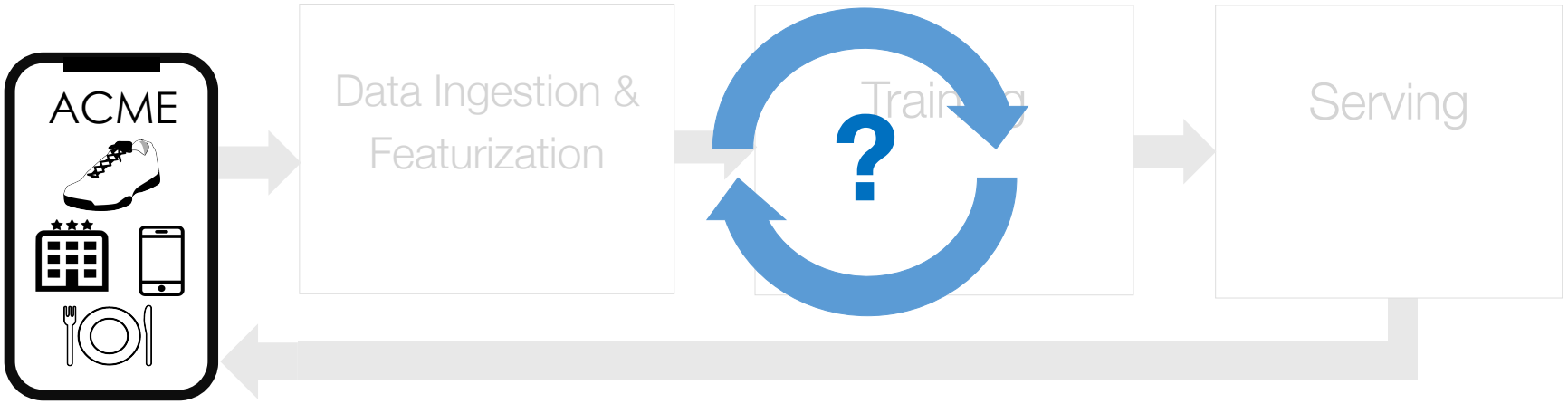
Example: In-app promotion

A **real** use case:

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

Example: In-app promotion

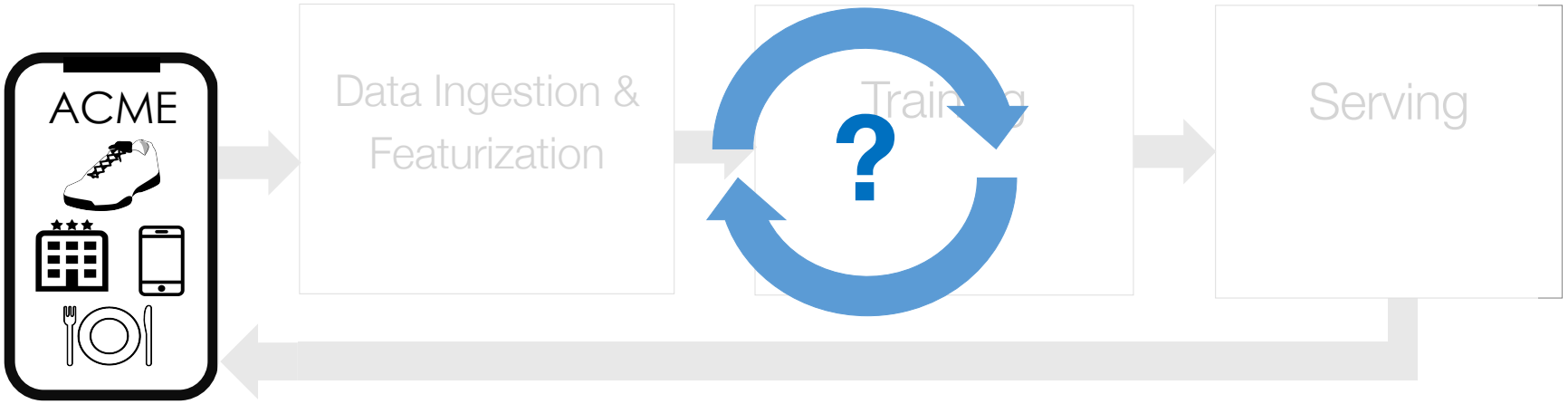
Example: In-app promotion



Two questions:

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

Example: In-app promotion



Model updated every **1 day**

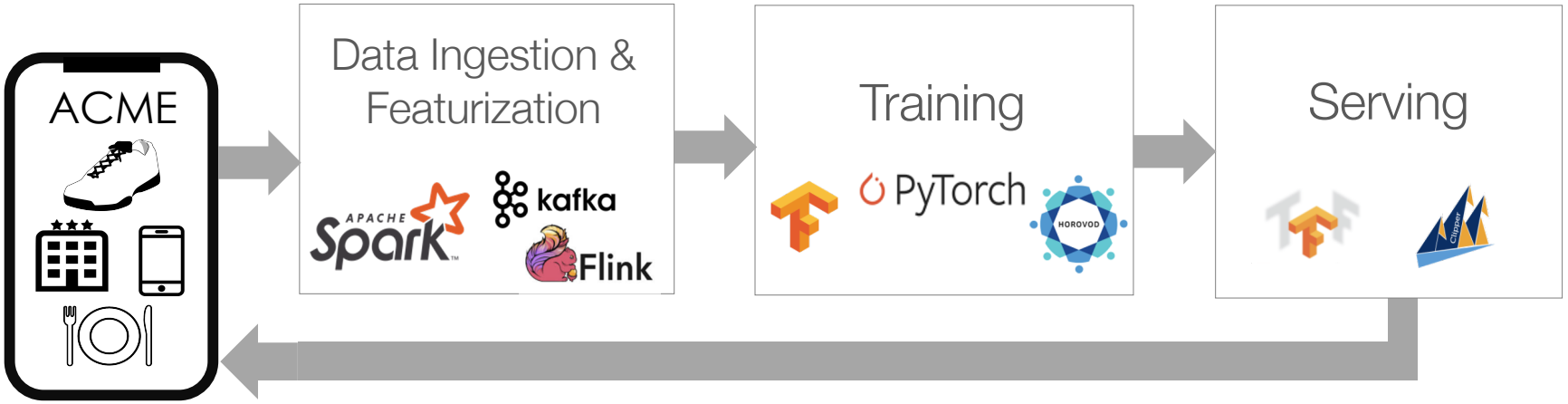
⇩ **+ 5%** CTR (Click Through Rate)

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

⇩

Want to get lower, but how?

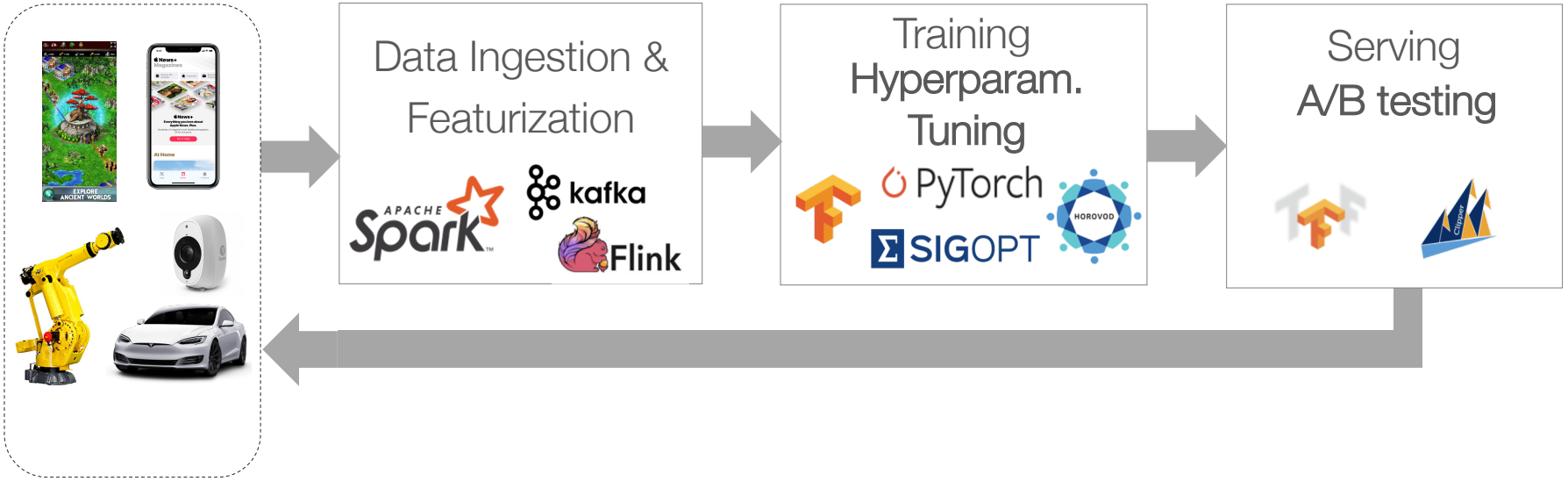
Example: In-app promotion



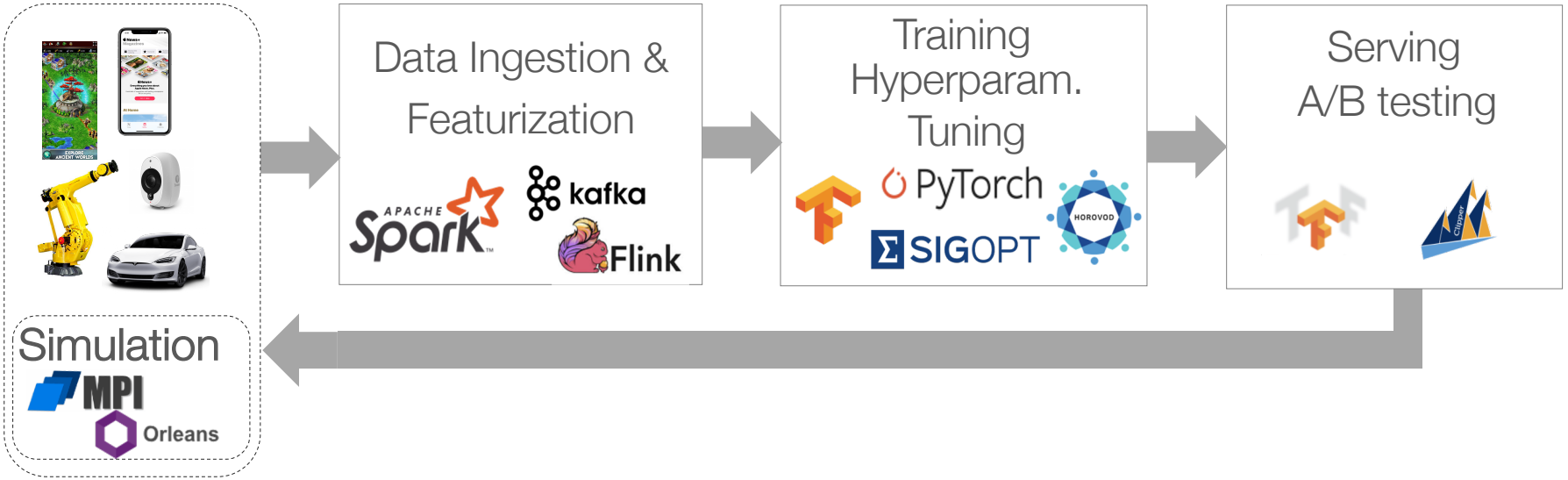
Previous solution: integrate best-of-breed frameworks

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

Even more complex patterns!



Even more complex patterns!



Reinforcement Learning

Today's ML Ecosystem

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	TensorFlow, Clipper	SIGOPT, talos AUTONOMY	Apache Spark, kafka, Flink	MPI, Orleans	Apache Spark, hadoop

Libraries

Training	Model Serving	Hyperparam. Tuning	Streaming	Simulation	Featurization

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

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

Three key ideas

Explicit parallelism: Execute remotely functions as tasks and instantiate remotely classes as actors

- **Support both stateful and stateless computations**

Asynchronous execution using futures

- **Enable parallelism**

Distributed (immutable) object store

- **Efficient communication** (send arguments by reference)

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)
```

Class

```
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

```
@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)
```

Class → Actor

```
@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

```
@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)
```

Class → Actor

```
@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()
```

Function → Task

```
@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)
```

Class → Actor

```
@ray.remote(num_gpus = 2)
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()
```

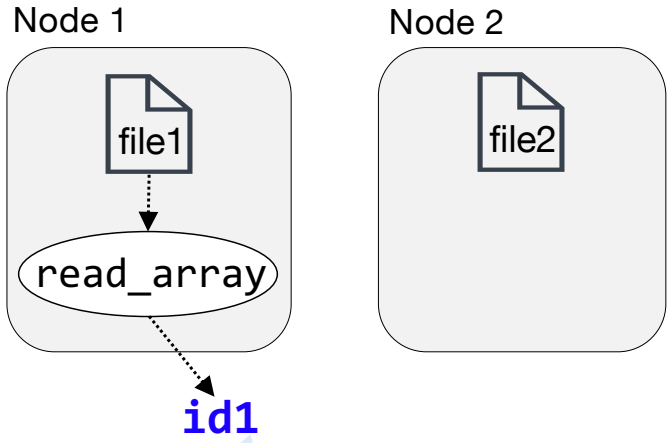
Task API

```
@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 (future) immediately, before `read_array()` finishes

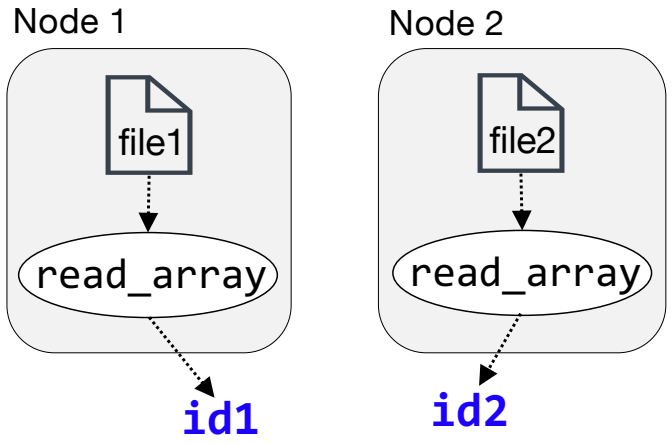
Task API

```
@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

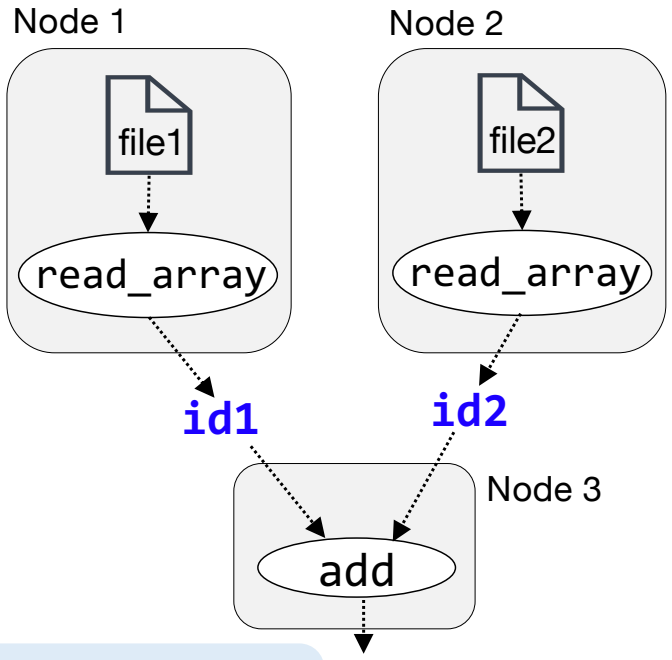
Task API

```
@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

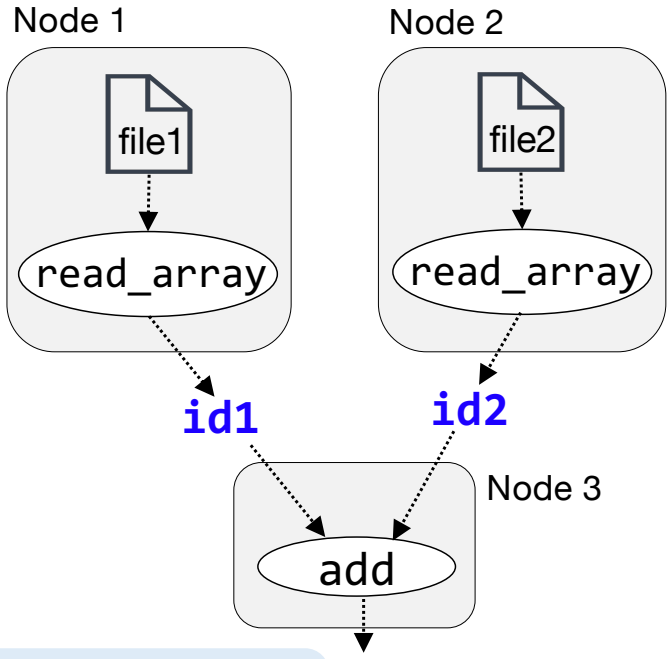
Task API

```
@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)
```

- Blue variables are Object IDs
- Similar to futures



`ray.get()` block until result available

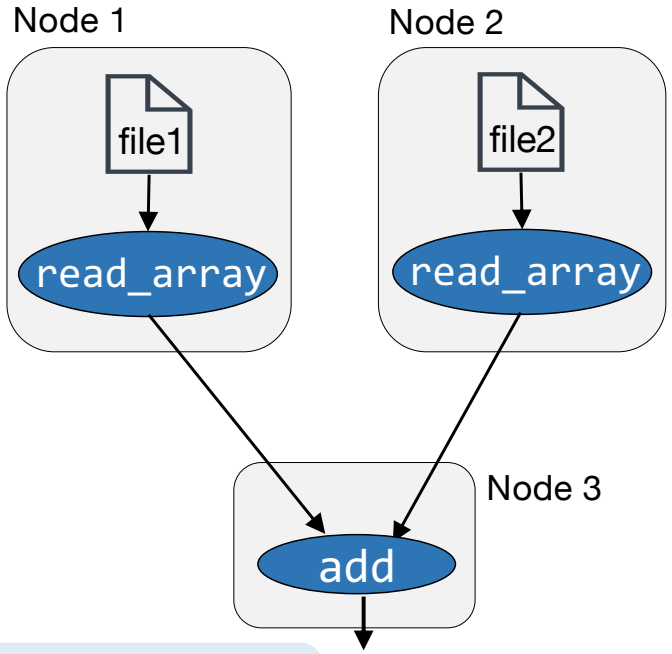
Task API

```
@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`

Minimalistic API: Core Ray API

API	Comments	Example
`ray.init()`	Initialize Ray context.	

`@ray.remote`	Function or class decorator specifying that the function will be executed as a task or the class as an actor in a different process.	`@ray.remote`
		`def fun(x):`
		`  …`
		
		`@ray.remote`
		`class Actor(object):`
		`  def method(y)`
		`  …`

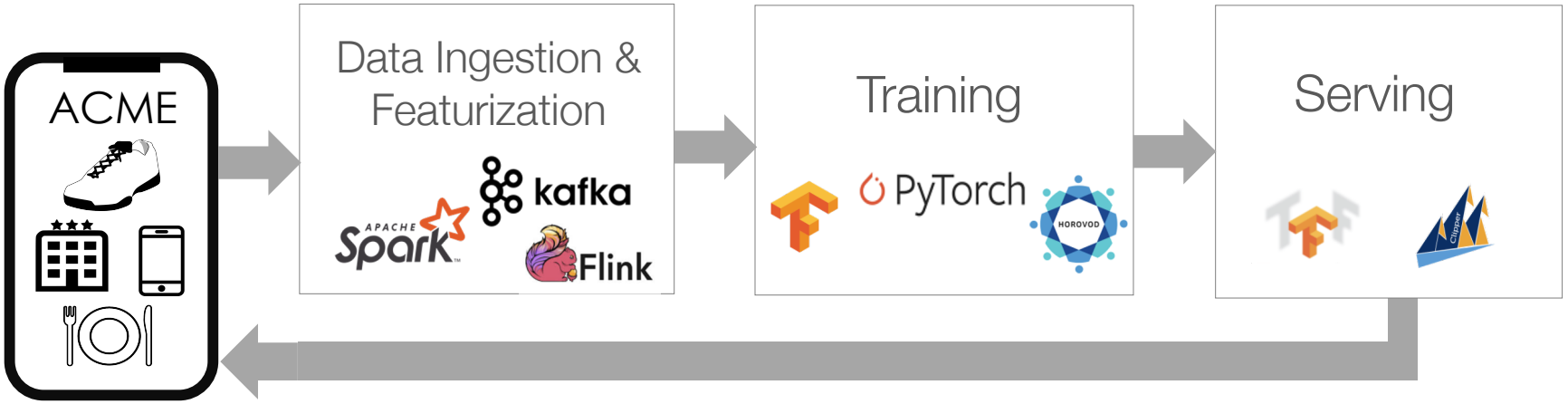
`.remote`	Postfix to every remote function, remote class declaration. Remote operations are asynchronous.	`ret_id = fun.remote(x)`
		`a = Actor.remote()`
		`ret_id = a.method.remote(y)`

`ray.put()`	Store object in object store, and return its ID.	`x_id = ray.put(x)`

`ray.get()`	Return object or list of objects from object ID or list of object IDs. This is a synchronous operation.	`x = ray.get(x_id)`
		`…`
		`objects = ray.get(object_ids)`

`ray.wait()`	From a list of object IDs returns (1) the list of IDs of the objects that are ready, and (2) the list of IDs of the objects that are not ready yet.	`ready_ids, not_ready_ids =` `ray.wait(object_ids)`

Example: In-app promotion



Model updated every **1 day**

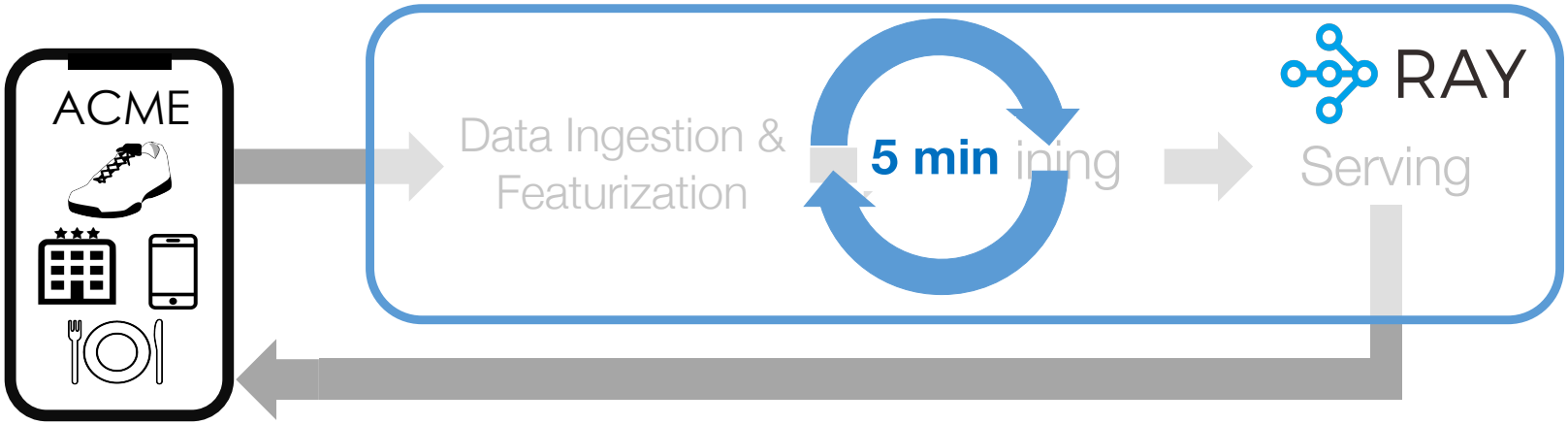
⇩ **+ 5%** CTR

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

⇩

?

Ray: unified platform for distributed apps



Model updated every **1 day**

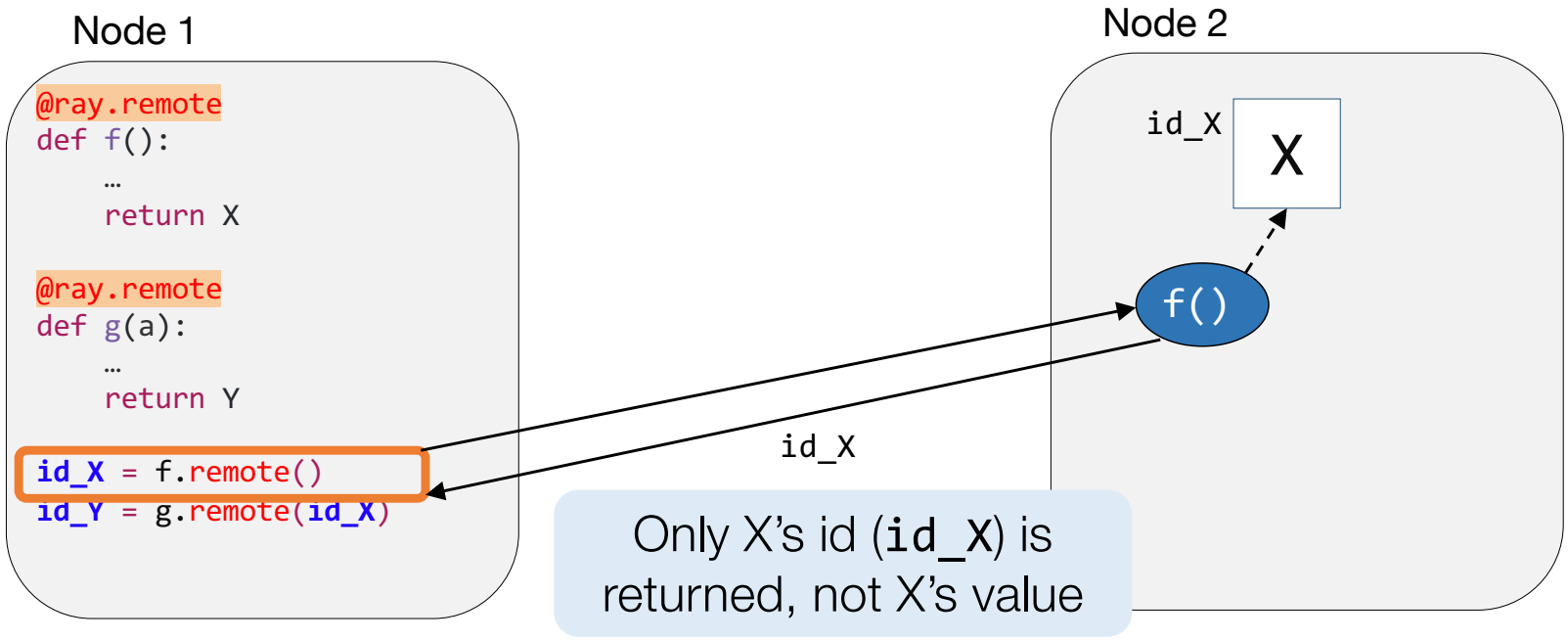
+ 5% CTR

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

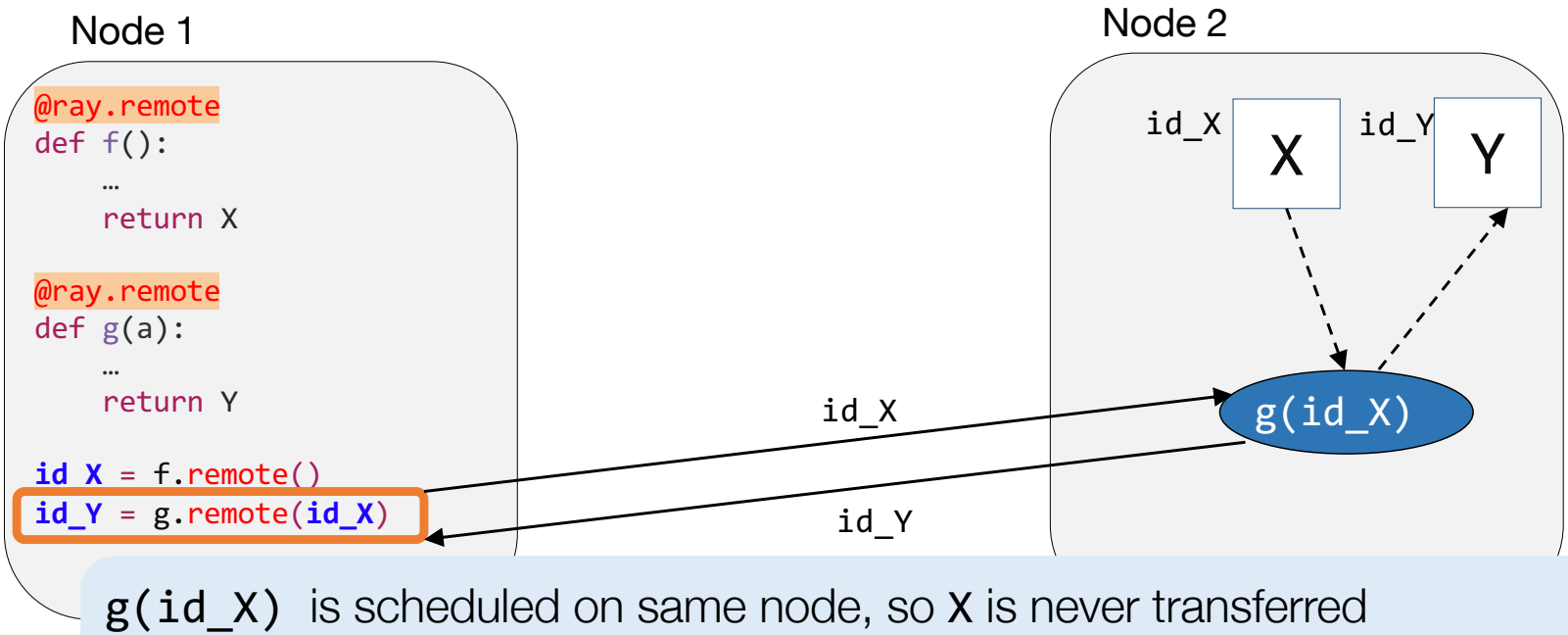
+ 1% CTR

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

Distributed object store



Distributed object store



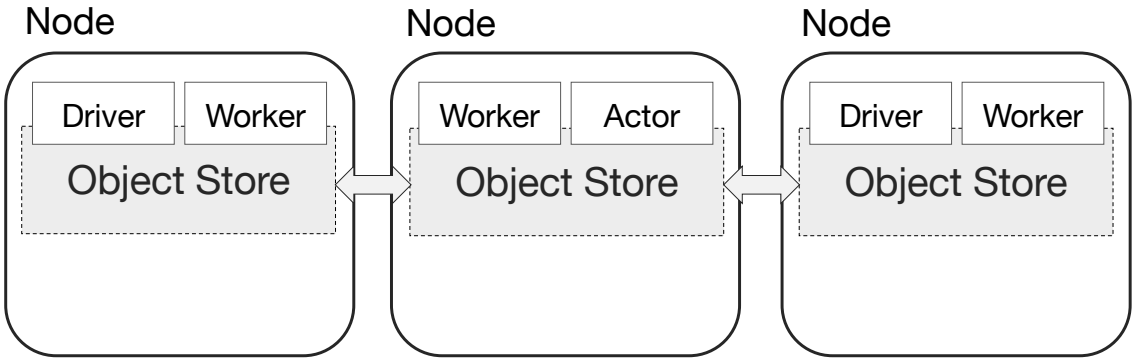
`g(id_X)` is scheduled on same node, so `X` is never transferred

Note: withot shared memory, X would have been transferred back and for from Node 1 and Node 2

Ray Architecture

Distributed object store

- Immutable objects

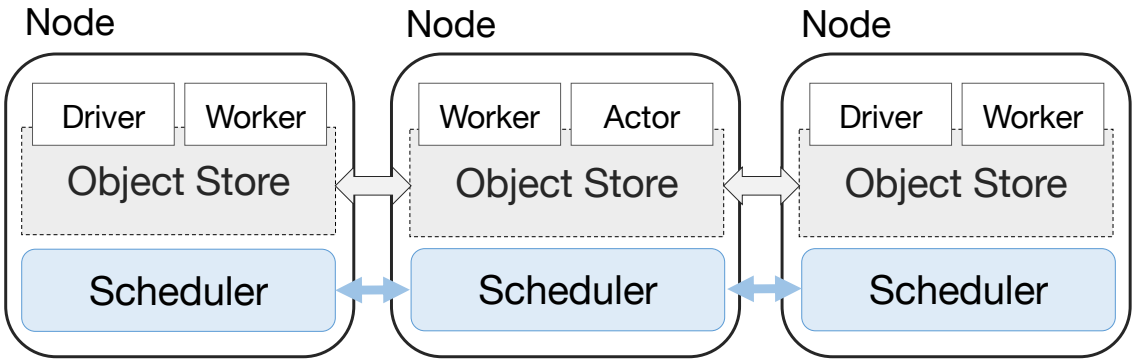


Ray Architecture

Distributed object store

- Immutable objects

Distributed scheduler



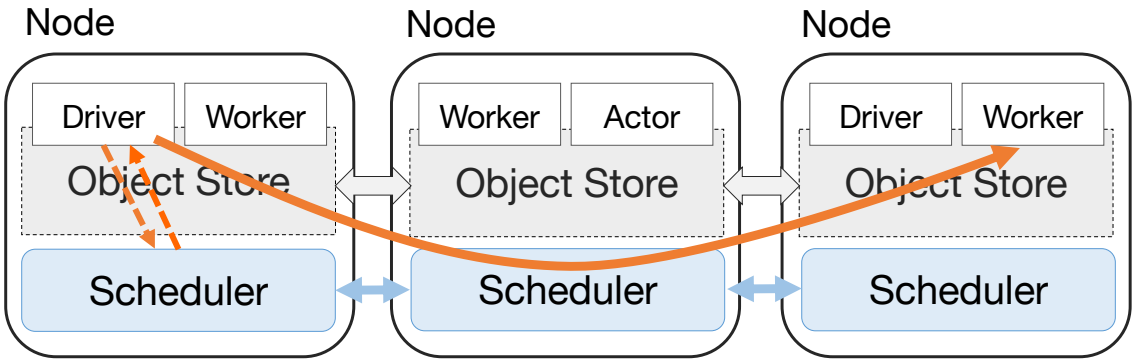
Ray Architecture

Distributed object store

- Immutable objects

Distributed scheduler

- Direct calls



Ray Architecture

Distributed object store

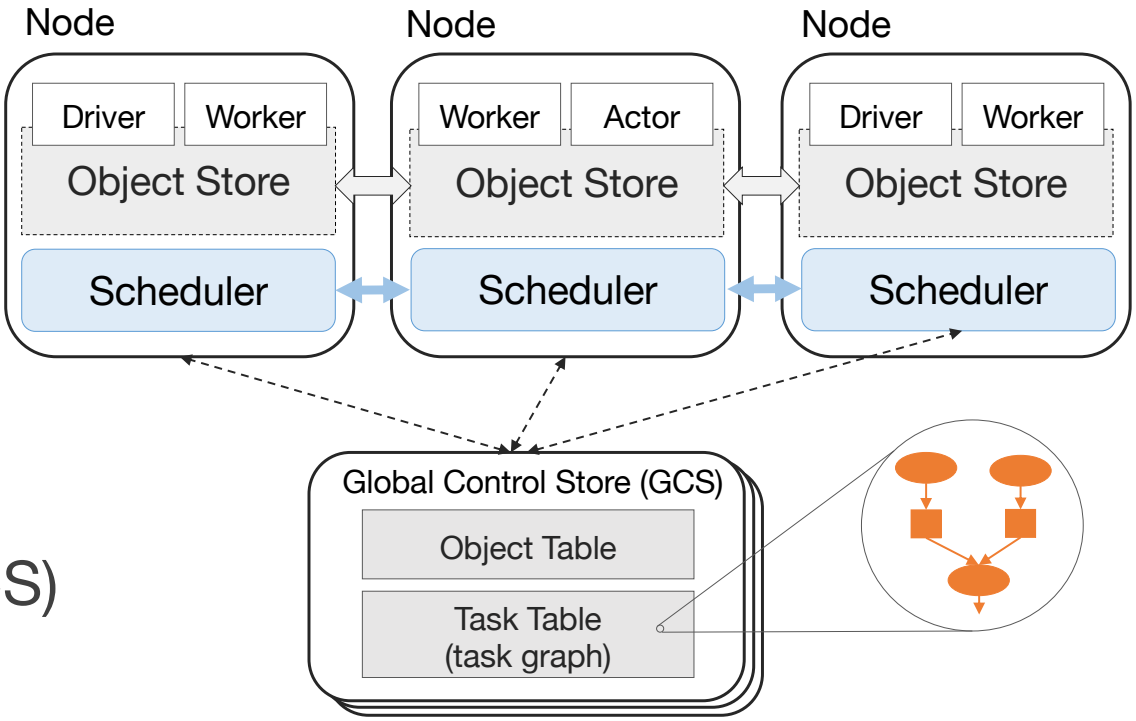
- Immutable objects

Distributed scheduler

- Direct calls

Central control store (GCS)

- Stateless components



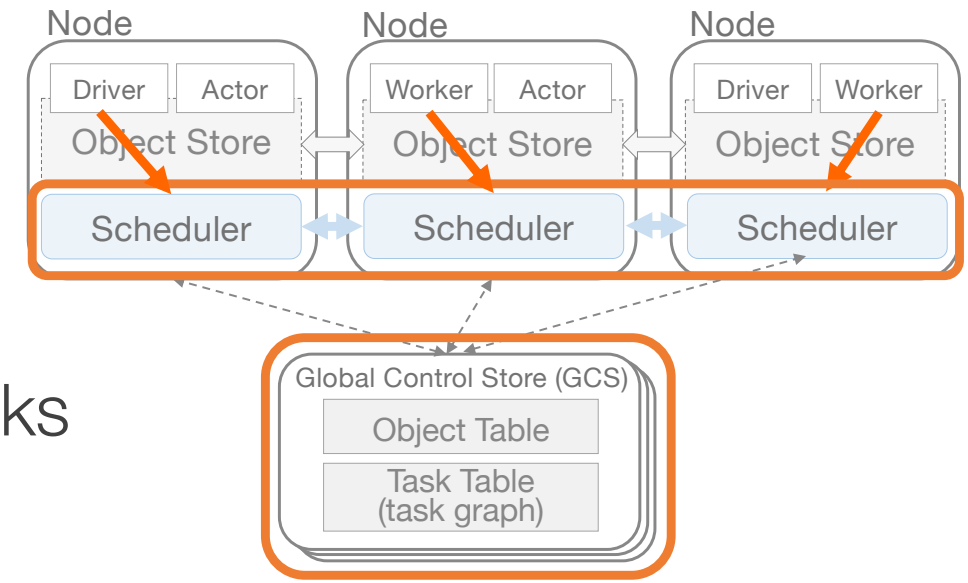
Horizontal scalaability

Decentralized scheduler

Sharded GCS

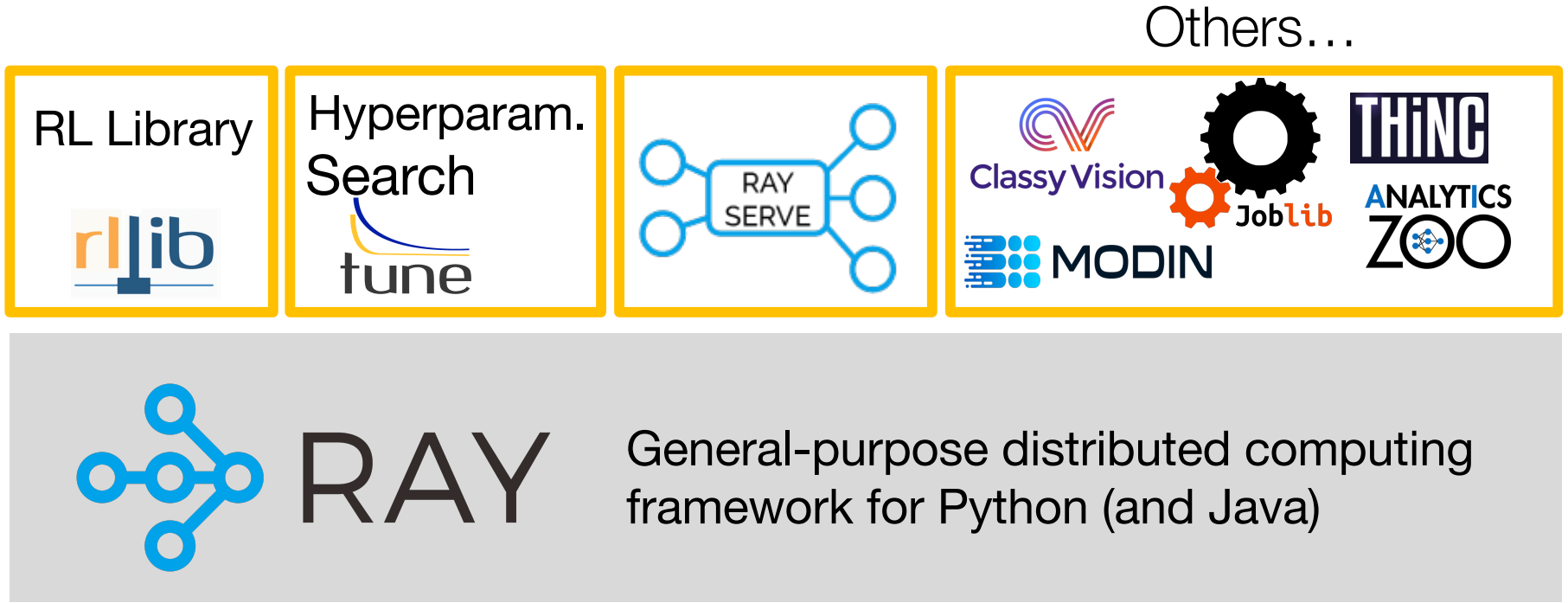
Any worker can submit tasks

- Driver not a bottleneck



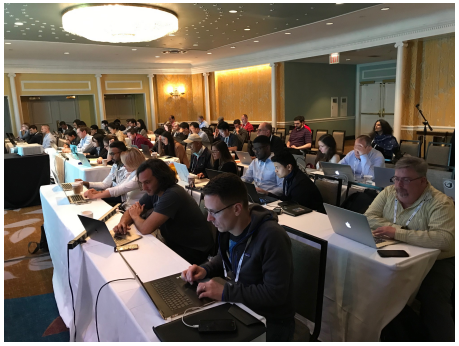
Ray vs specialized systems

Match performance of specialized systems


Others…
RL Library
rllib
Hyperparam.
Search
tune
RAY
SERVE
Classy Vision
Joblib
THiNC
ANALYTICS
ZOO
MODIN
RAY
General-purpose distributed computing
framework for Python (and Java)

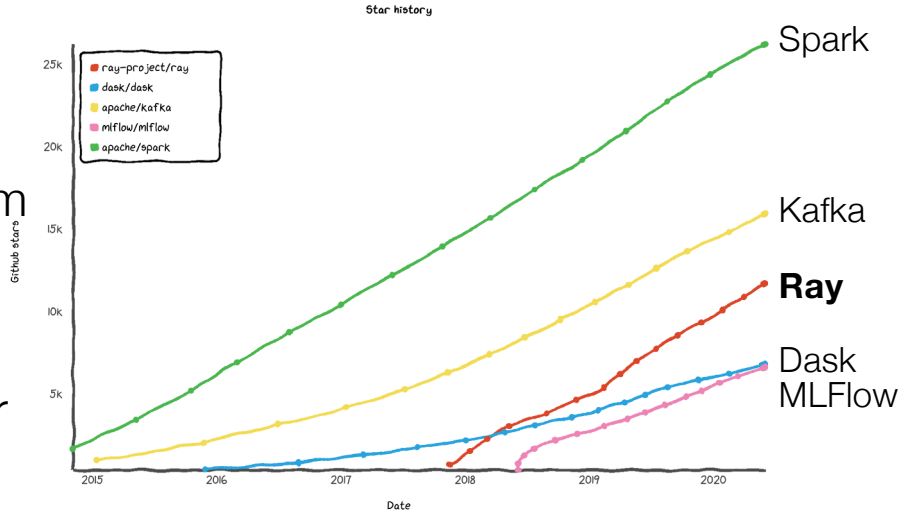
https://github.com/ray-project/ray

Growing adoption



~12K github stars
300+ contributors from
50+ companies

Included in Azure ML,
and AWS Sage Maker



蚂蚁金服 ANT FINANCIAL
Alibaba Group
intel
J.P.Morgan
Morgan Stanley
facebook
amazon
PRIMER
ERICSSON
Microsoft
bonsai

Ray summary

Ray ~= RPC + Actors + Shared memory

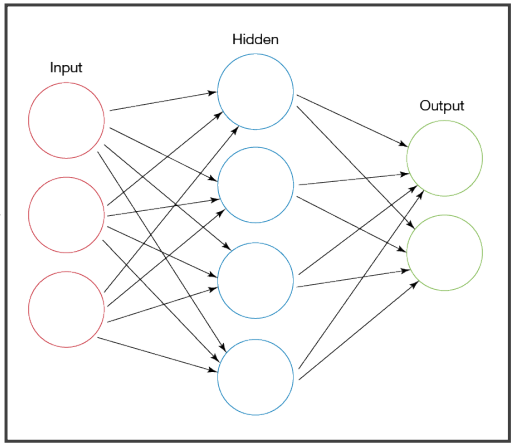
Possible API for future serverless

- Allows developers to specify resources for each computation
- Enables systems to optimize resource allocation

- Ray

- Neurocuts
- Autopandas

“Classic” DL/RL apps



Speech recognition

Video recognition

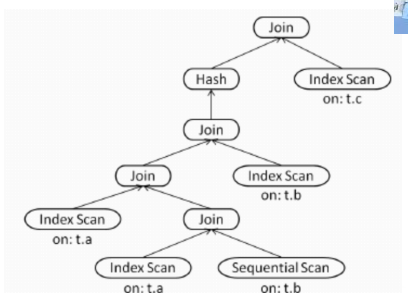
Language translation

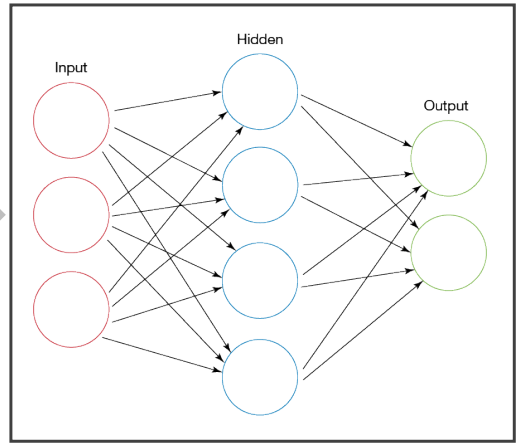
…

Human tasks: 100% accuracy not expected

Systems problems

```
SELECT COALESCE(users.state,'') AS "_g1",
       users.state AS `users.state
       COUNT(DISTINCT orders.id) A
FROM orders
LEFT JOIN users ON orders.user_id =
```





Program synthesis

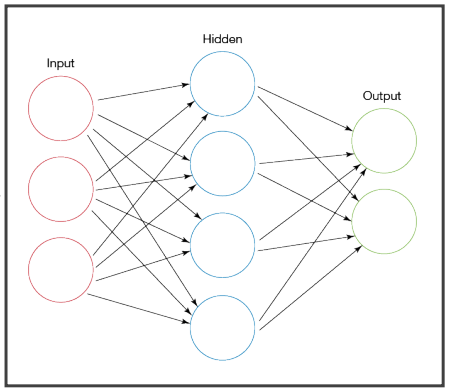
Mortgage decisions

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

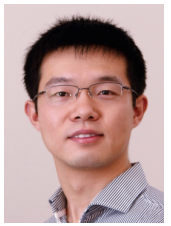
Generate & Verify

Generate solutions whose correctness can be verified

- Ray

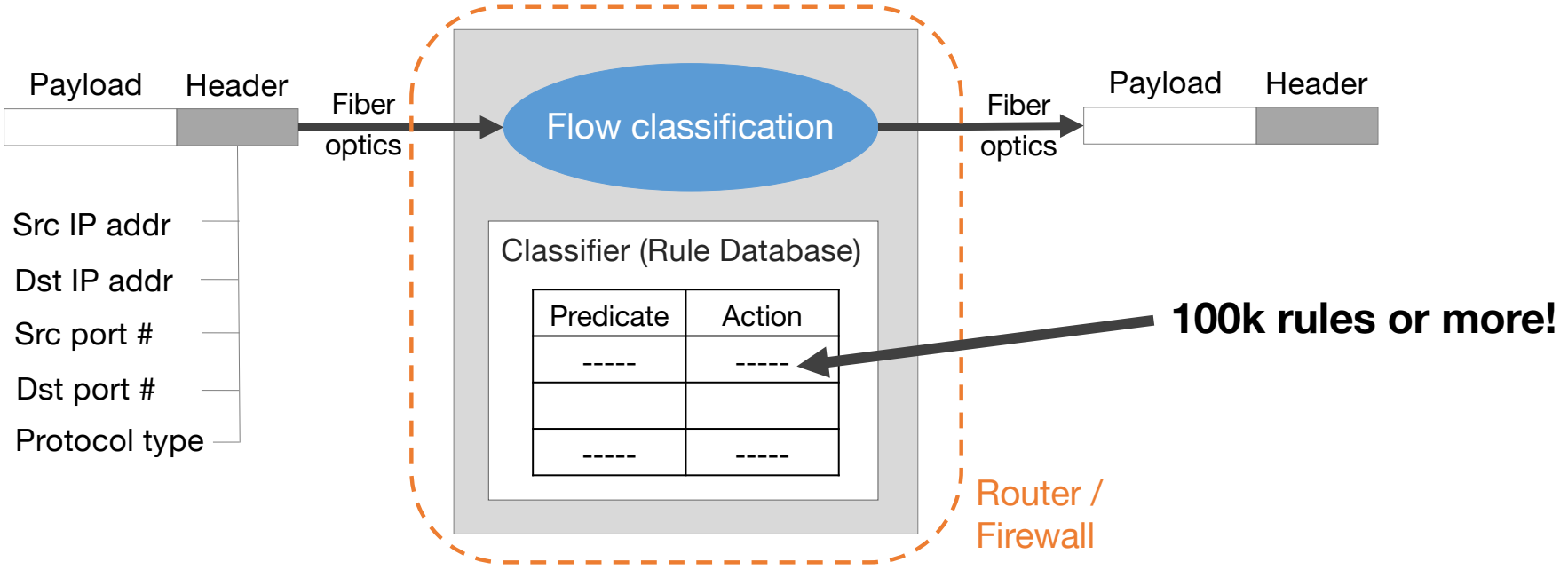
- **Neurocuts**
- Autopandas

Packet Classification




Fundamental problem in networking

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

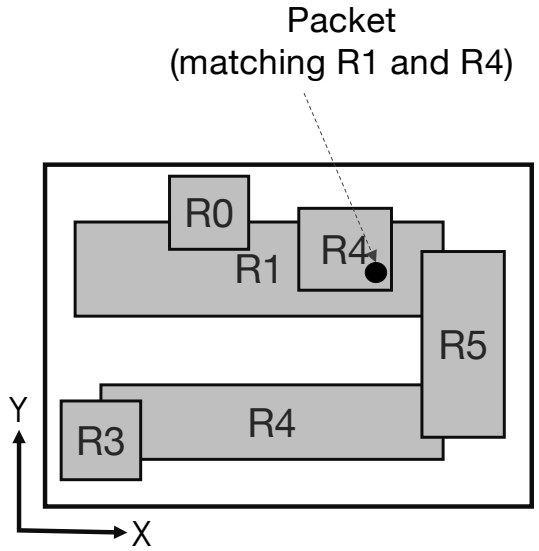


The problem

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

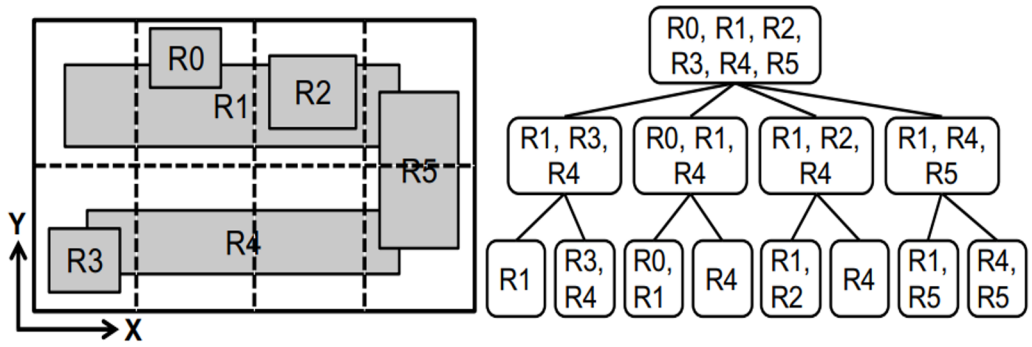
20+ years of work

Hardware

- Expensive and power hungry → prohibitive for large classifiers

Software

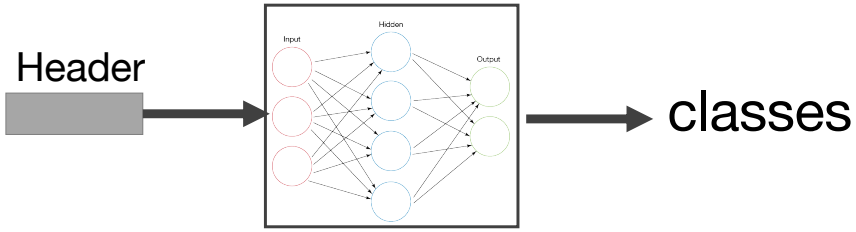
- Build a multi-dimensional "decision tree" – really a k-d index
- HiCuts ('99), HyperCuts ('03), EffiCuts ('10), CutSplit ('15)
  - All rely on hand-tuned heuristics, which are brittle and not optimal



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

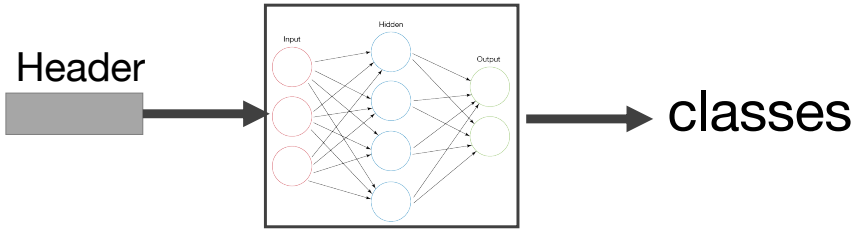
Two approaches

1. End-to-end solution

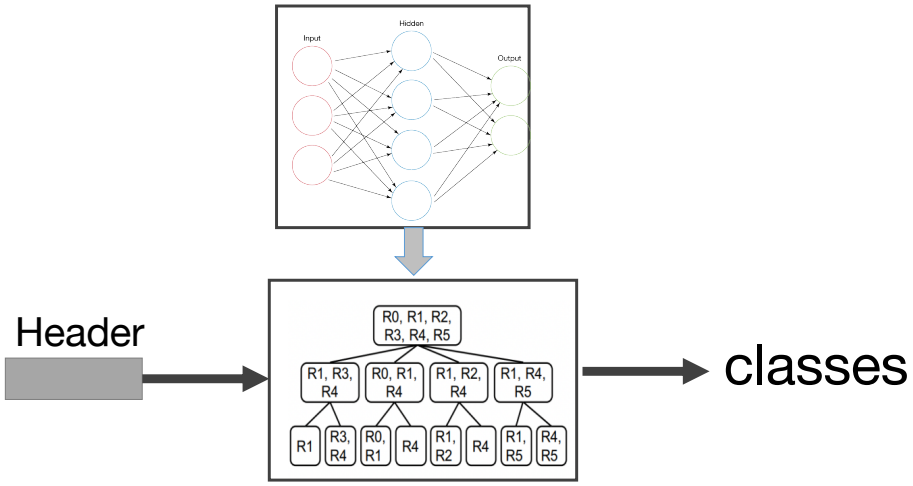


Two approaches

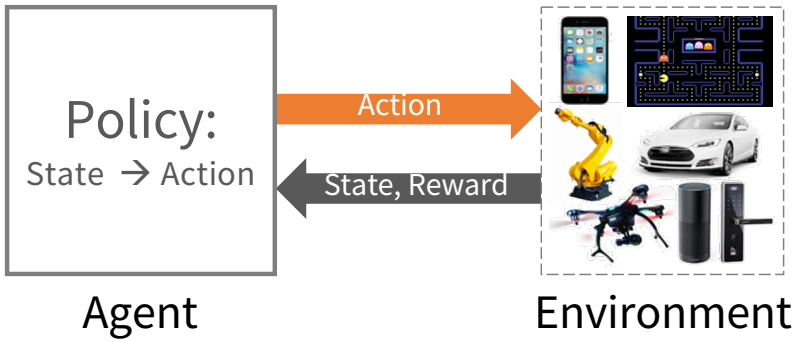
1. End-to-end solution



2. Build classification tree



Reinforcement Learning (RL)



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

NeuroCuts*: Building decision trees with Deep RL

RL formulation:

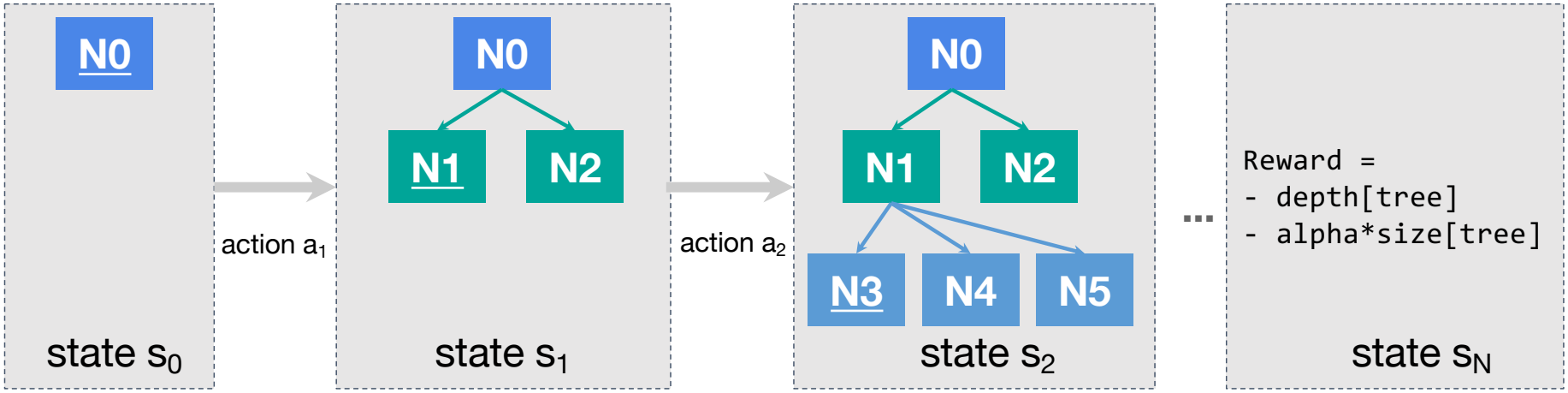
- **State**: intermediate decision tree
- **Action**: split or replicate a node
- **Reward**: negative of tree depth and tree size

*” Neural Packet Classification”, Eric Liang et al., SIGCOMM ‘19

NeuroCuts*: Building decision trees with Deep RL

RL formulation:

- **State**: intermediate decision tree
- **Action**: split or replicate a node
- **Reward**: negative of tree depth and tree size

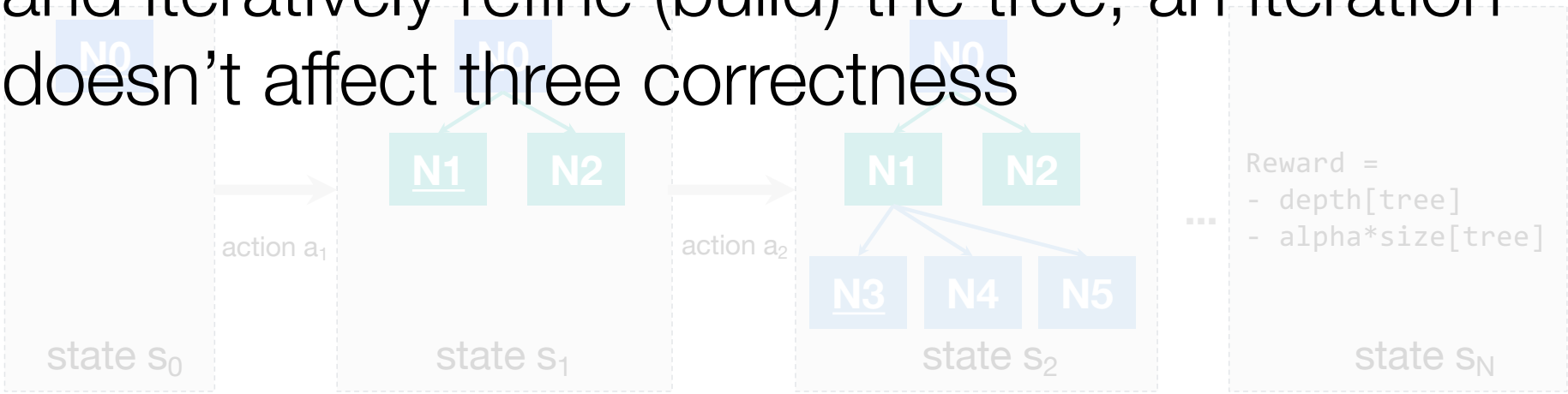


NeuroCuts*: Building decision trees with Deep RL

RL formulation:

- State: intermediate decision tree
- Action: split/partition a node
- Reward: negative of tree depth and tree size

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



Results

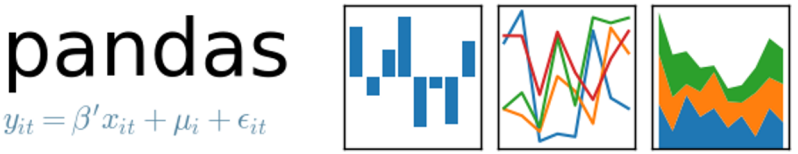
Classification time: median 18% faster than state-of-the-art

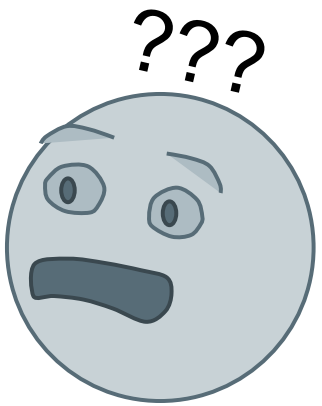
3x better either in space or time than any previous solution

- Ray

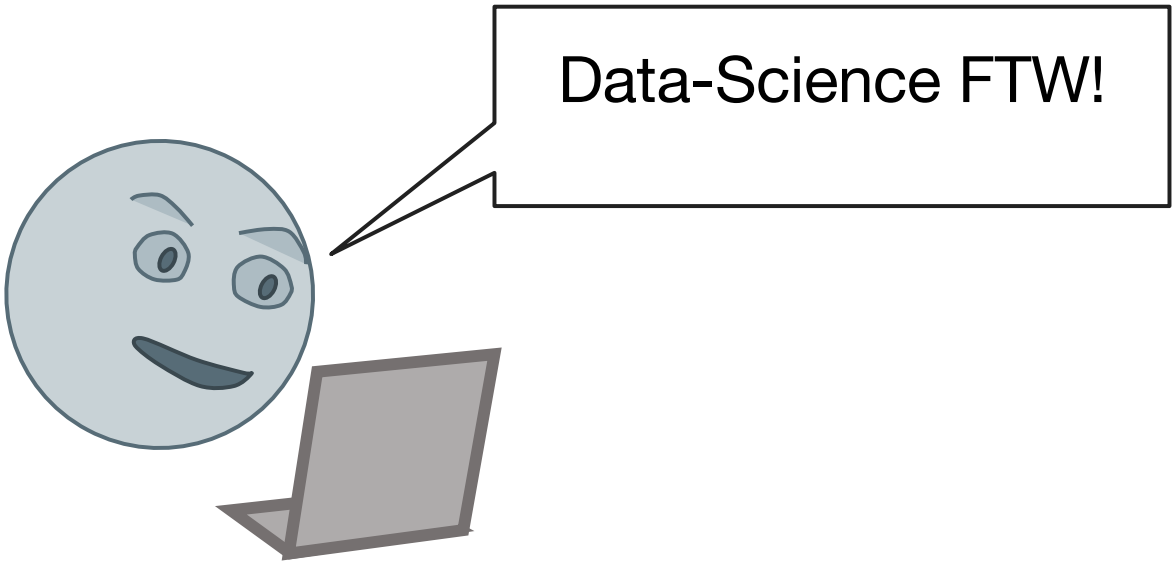
- Neurocuts
- **Autopandas**

API Explosion!

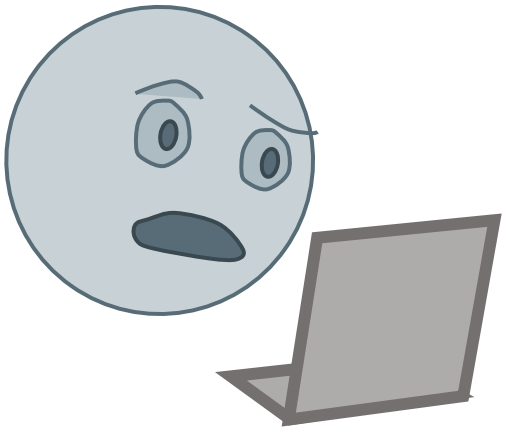




Difficult to keep up with APIs



Difficult to keep up with APIs



DataFrame

Constructor

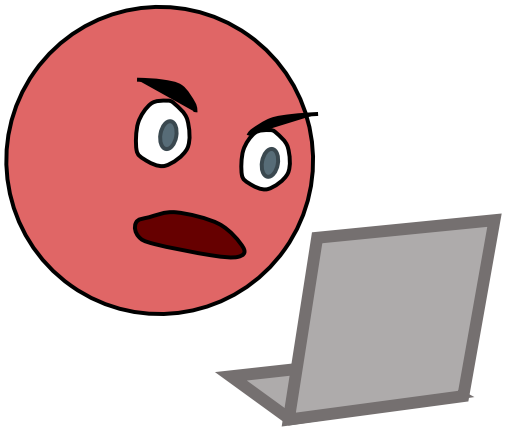
`DataFrame`([data, index, columns, dtype, copy])	Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

Attributes and underlying data

Axes

`DataFrame.index`	The index (row labels) of the DataFrame.
`DataFrame.columns`	The column labels of the DataFrame.
`DataFrame.dtypes`	Return the dtypes in the DataFrame.
`DataFrame.ftypes`	(DEPRECATED) Return the ftypes (indication of sparse/dense and dtype) in DataFrame.
`DataFrame.get_dtype_counts`(self)	(DEPRECATED) Return counts of unique dtypes in this object.
`DataFrame.get_ftype_counts`(self)	(DEPRECATED) Return counts of unique ftypes in this object.
`DataFrame.select_dtypes`(self[, include, exclude])	Return a subset of the DataFrame's columns based on the column dtypes.
`DataFrame.values`	Return a Numpy representation of the DataFrame.
`DataFrame.get_values`(self)	(DEPRECATED) Return an ndarray after converting sparse values to dense.
`DataFrame.axes`	Return a list representing the axes of the DataFrame.
`DataFrame.ndim`	Return an int representing the number of axes / array

Difficult to keep up with APIs



pandas.DataFrame.pivot_table

`DataFrame.pivot_table(self, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All', observed=False)` [source]

Create a spreadsheet-style pivot table as a DataFrame. The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.

Parameters:

values : *column to aggregate, optional*

index : *column, Grouper, array, or list of the previous*
If an array is passed, it must be the same length as the data. The list can contain any of the other types (except list). Keys to group by on the pivot table index. If an array is passed, it is being used as the same manner as column values.

columns : *column, Grouper, array, or list of the previous*
If an array is passed, it must be the same length as the data. The list can contain any of the other types (except list). Keys to group by on the pivot table column. If an array is passed, it is being used as the same manner as column values.

aggfunc : *function, list of functions, dict, default numpy.mean*
If list of functions passed, the resulting pivot table will have hierarchical columns whose top level are the function names (inferred from the function objects themselves) If dict is passed, the key is column to aggregate and value is function or list of functions

fill_value : *scalar, default None*
Value to replace missing values with

margins : *boolean, default False*
Add all row / columns (e.g. for subtotal / grand totals)

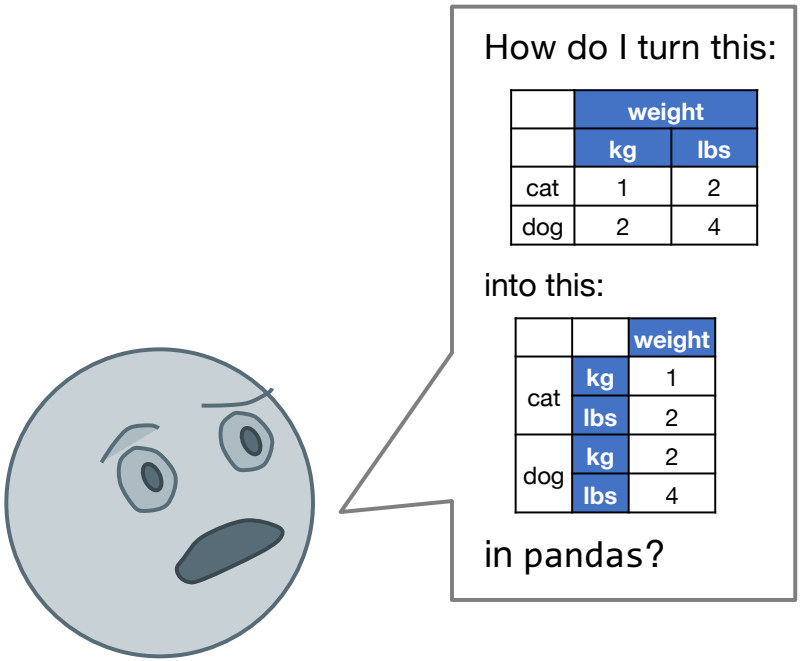
dropna : *boolean, default True*
Do not include columns whose entries are all NaN

margins_name : *string, default 'All'*
Name of the row / column that will contain the totals when margins is True.

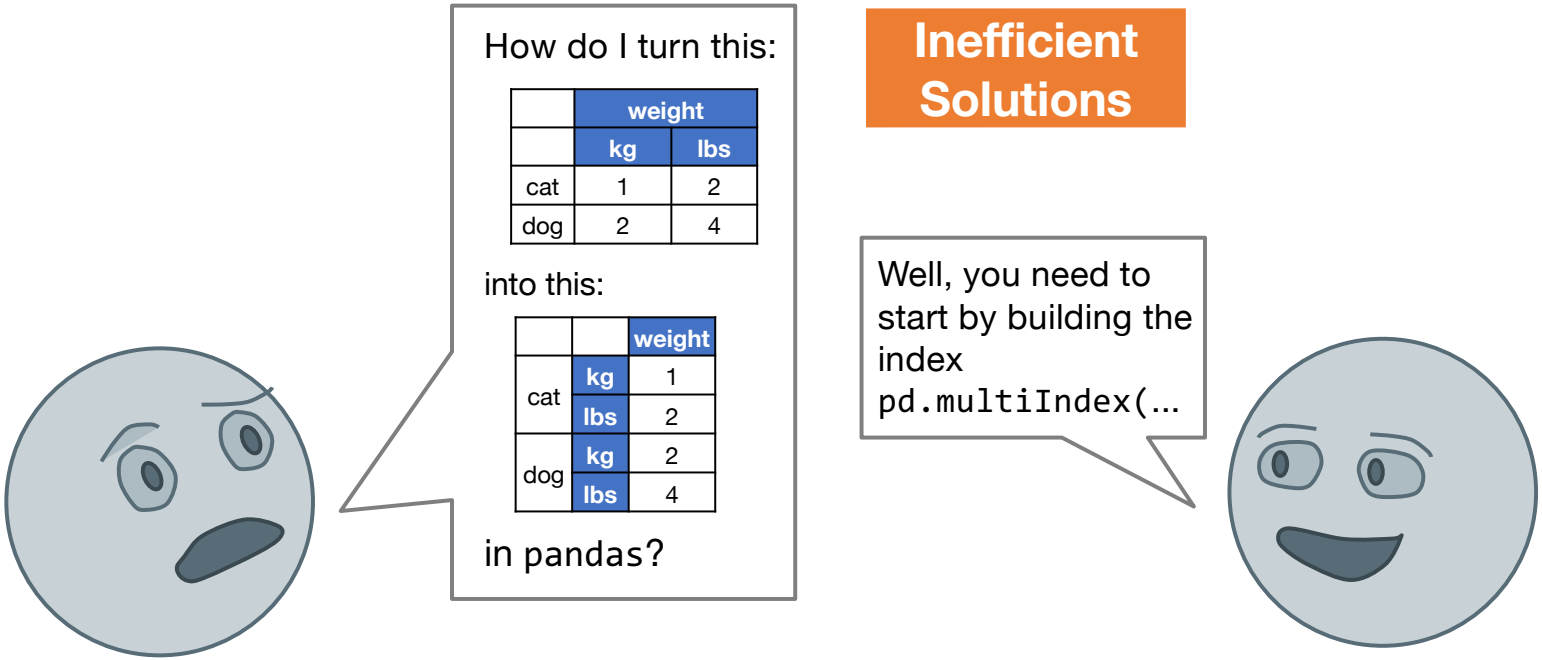
observed : *boolean, default False*
This only applies if any of the groupers are Categoricals. If True: only show observed values for categorical groupers. If False: show all values for categorical groupers.
Changed in version 0.25.0.

Returns: DataFrame

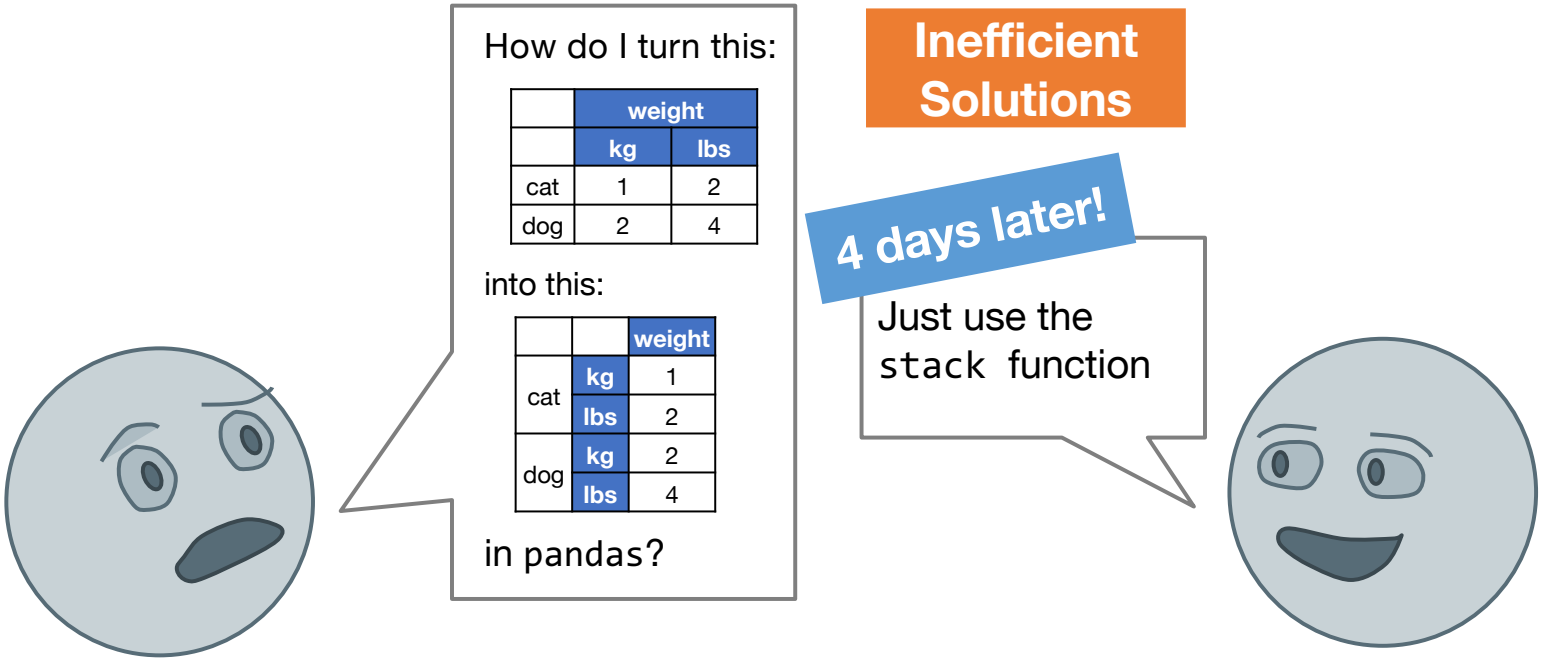
How to cope? *StackOverflow*



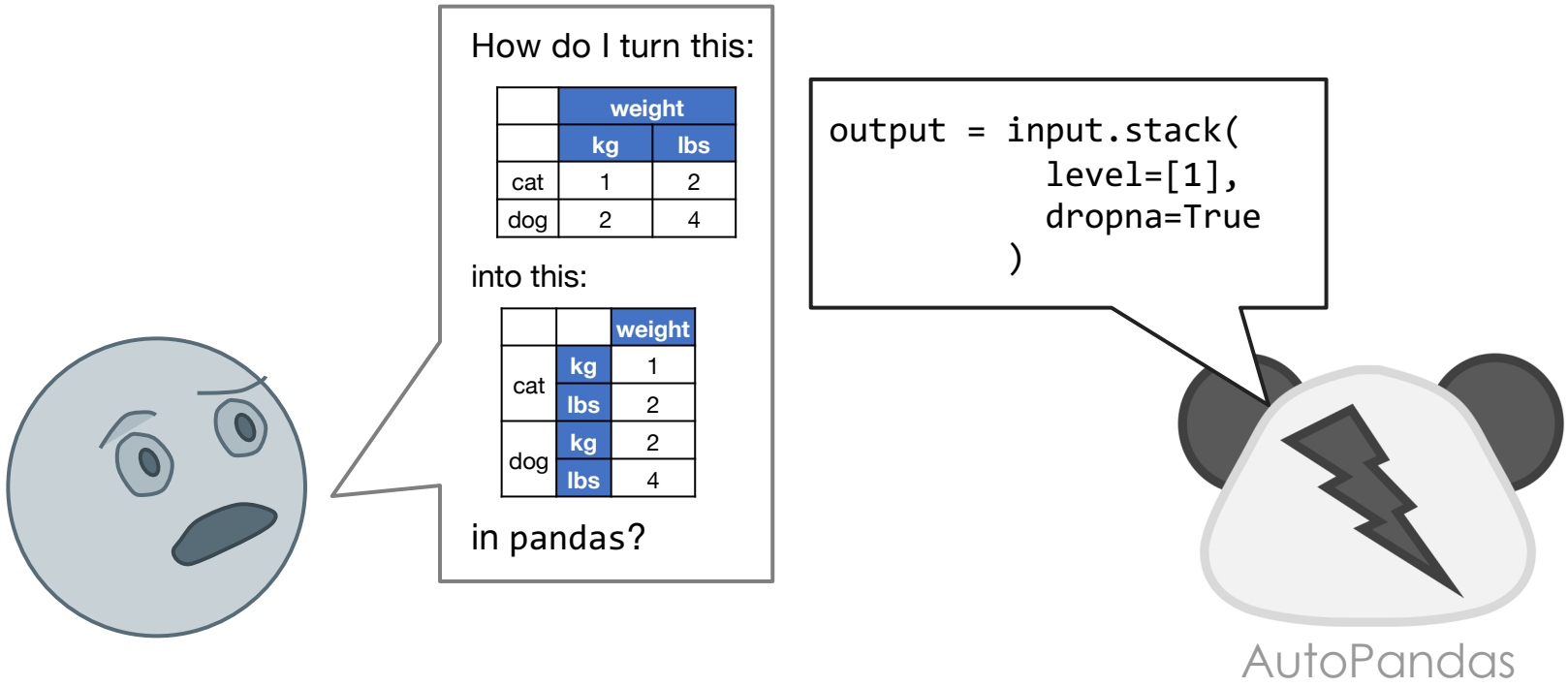
Problems with *StackOverflow*



Problems with *StackOverflow*



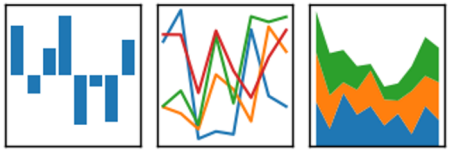
Goal: *StackOverflow* for APIs via Program Synthesis



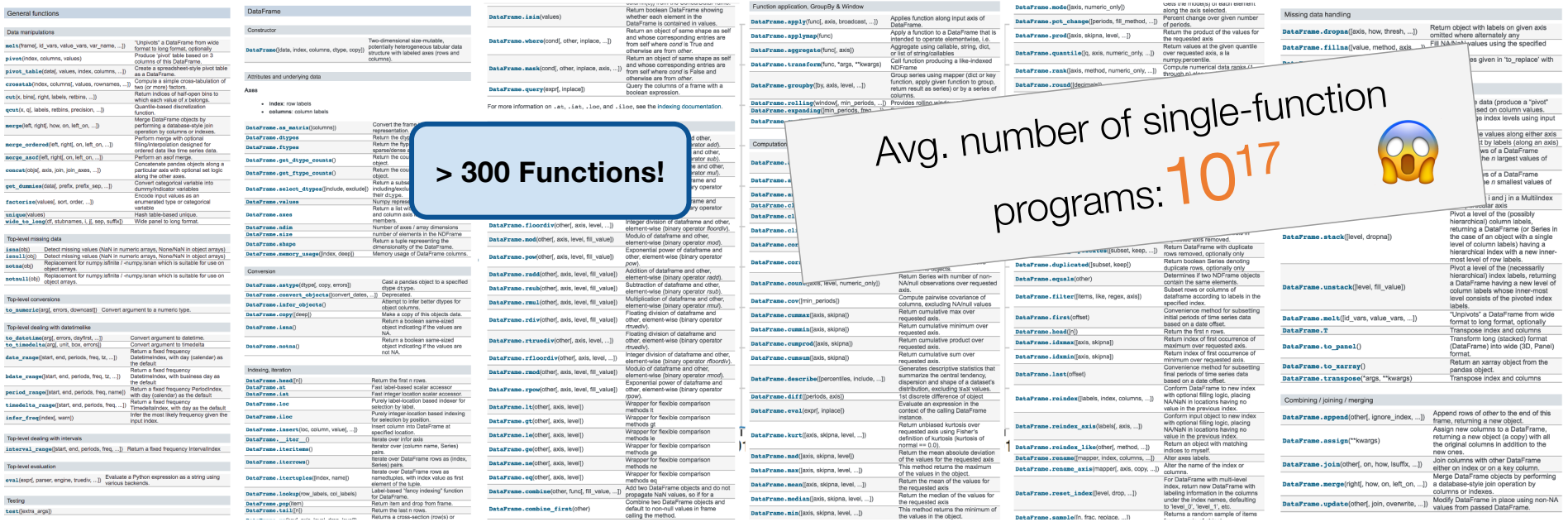
Challenges

pandas

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



Premier library for data scientists



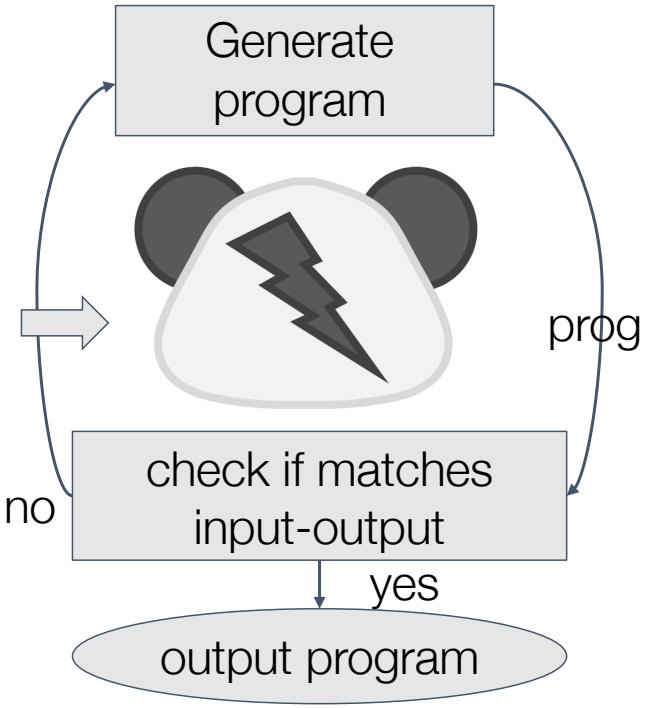
Autopandas*

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

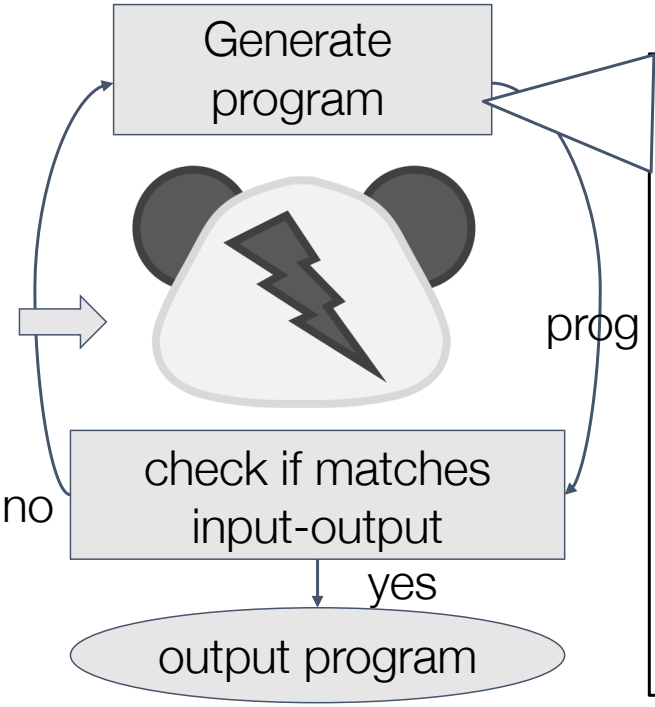
Predict program

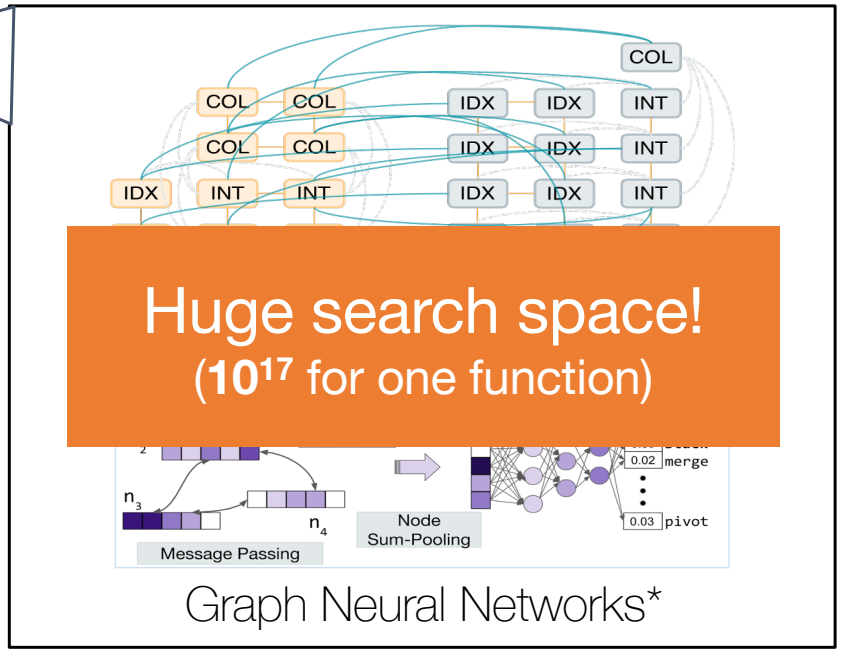
input:

	weight	
	kg	lbs
cat	1	2
dog	2	4

output:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4





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

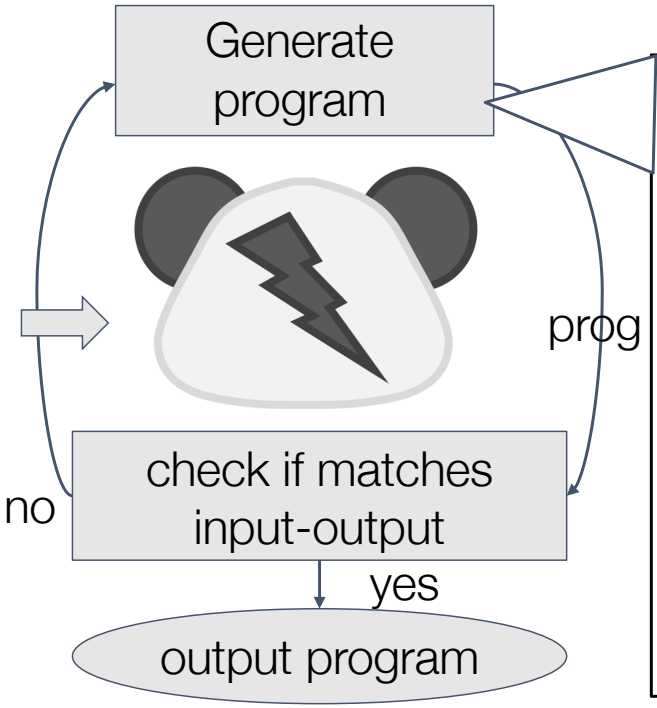
Neural-backed program generators

input:

	weight	
	kg	lbs
cat	1	2
dog	2	4

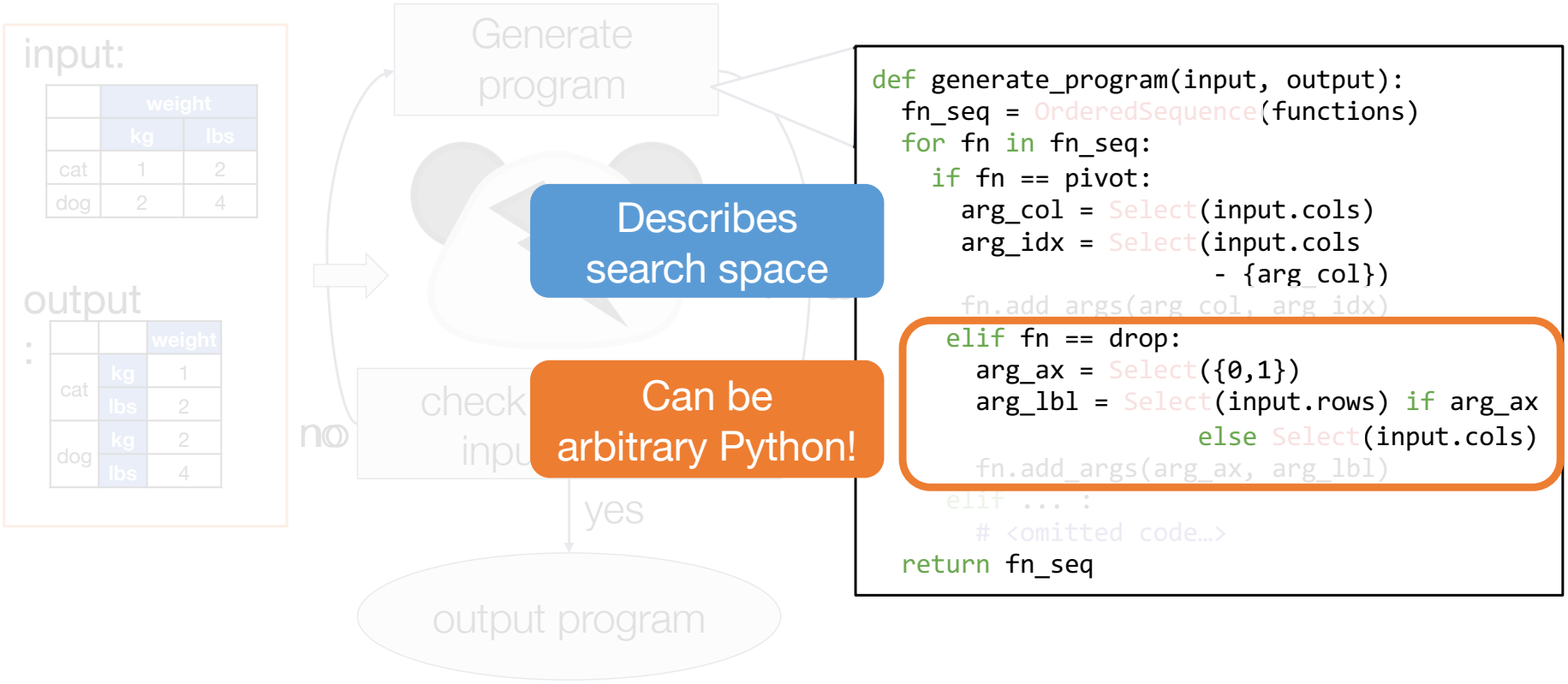
output:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4




```
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
```

Domain expertise to reduce search space



Neural-backed program generators

input:

	weight	
	kg	lbs
cat	1	2
dog	2	4

output:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4


```
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})

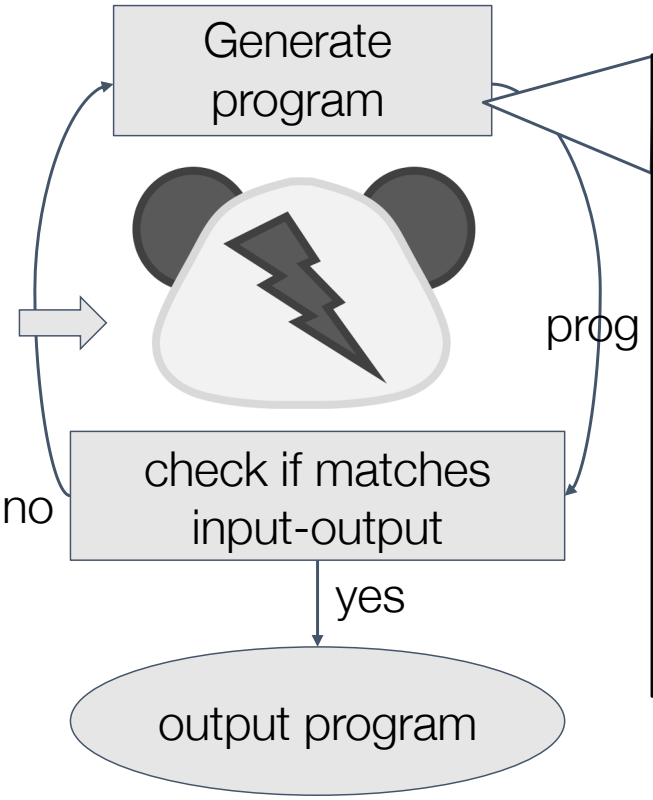
Neural-backed program generators

input:

	weight	
	kg	lbs
cat	1	2
dog	2	4

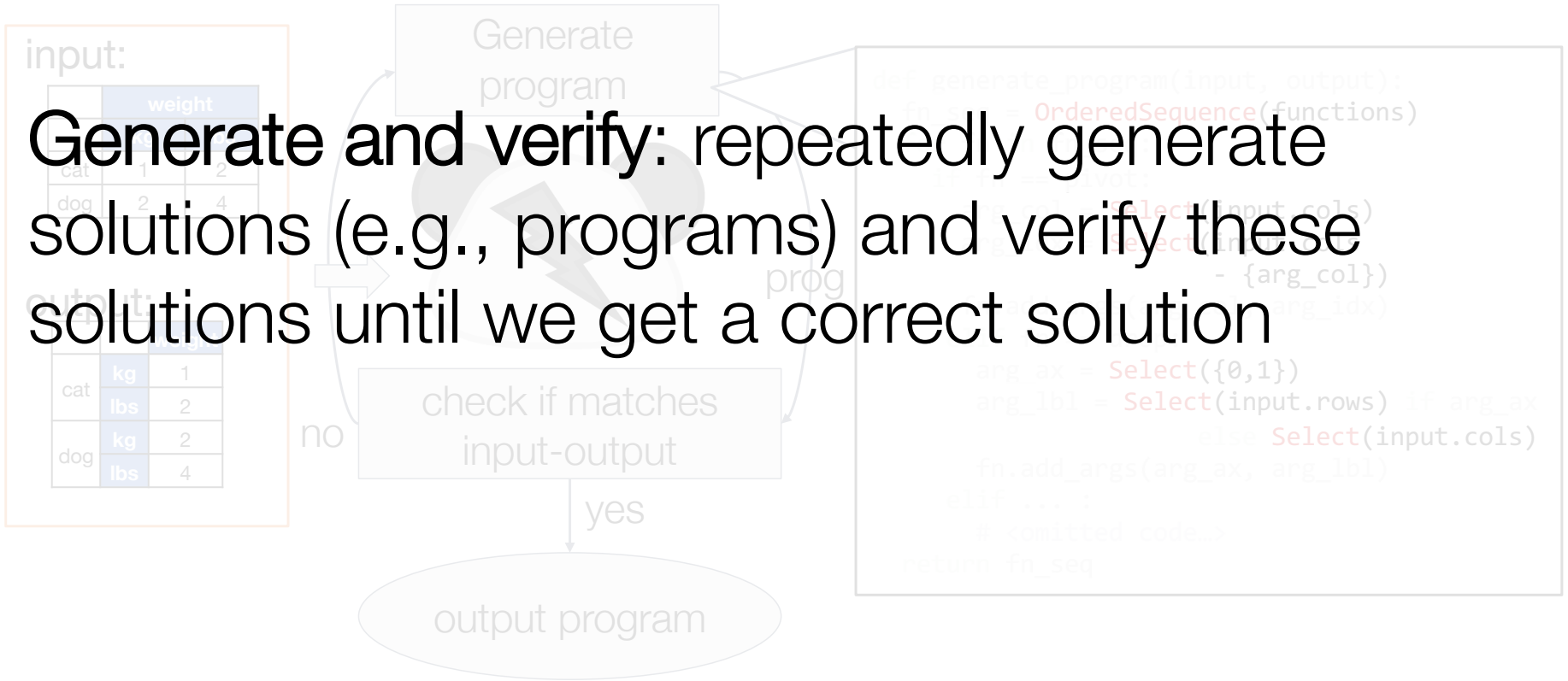
output:

		weight
cat	kg	1
	lbs	2
dog	kg	2
	lbs	4




```
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



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

Results on Real Stackoverflow Benchmarks

	Depth	Candidates Explored	Sequences Explored	Solved	Time(s)
SO_11881165	1	15	1	Y	0.54
SO_11941492	1	783	8	Y	12.55
SO_13647222	1	5	1	Y	3.32
SO_18172851	1	-	-	N	-
SO_49583055	1	-	-	N	-
SO_49592930	1	2	1	Y	1.1
SO_49572546	1	3	1	Y	1.1
SO_13261175	1	39537	18	Y	300.2
SO_13793321	1	92	1	Y	4.16
SO_14085517	1	10	1	Y	2.24
SO_11418192	2	158	1	Y	0.71
SO_49567723	2	1684022	2	Y	753.1
SO_13261691	2	65	1	Y	2.96
SO_13659881	2	2	1	Y	1.38
SO_13807758	2	711	2	Y	7.21
SO_34365578	2	-	-	N	-
SO_10982266	3	-	-	N	-
SO_11811392	3	-	-	N	-
SO_49581206	3	-	-	N	-
SO_12065885	3	924	1	Y	0.9
SO_13576164	3	22966	5	Y	339.25
SO_14023037	3	-	-	N	-
SO_53762029	3	27	1	Y	1.9
SO_21982987	3	8385	10	Y	30.8
SO_39656670	3	-	-	N	-
SO_23321300	3	-	-	N	-

Collected 26 Real-World Stack-Overflow Benchmarks

Encouraging Results on Real Stackoverflow Benchmarks

	Depth	Candidates Explored	Sequences Explored	Solved	Time(s)
SO_11881165	1	15	1	Y	0.54
SO_11941492	1	783	8	Y	12.55
SO_13647222	1	5	1	Y	3.32
SO_18172851	1	-	-	N	-
SO_49583055	1	-	-	N	-
SO_49592930	1	2	1	Y	1.1
SO_49572546	1	3	1	Y	1.1
SO_13261175	1	39537	18	Y	300.2
SO_13793321	1	92	1	Y	4.16
SO_14085517	1	10	1	Y	2.24
SO_11418192	2	158	1	Y	0.71
SO_49567723	2	1684022	2	Y	753.1
SO_13261691	2	65	1	Y	2.96
SO_13659881	2	2	1	Y	1.38
SO_13807758	2	711	2	Y	7.21
SO_34365578	2	-	-	N	-
SO_10982266	3	-	-	N	-
SO_11811392	3	-	-	N	-
SO_49581206	3	-	-	N	-
SO_12065885	3	924	1	Y	0.9
SO_13576164	3	22966	5	Y	339.25
SO_14023037	3	-	-	N	-
SO_53762029	3	27	1	Y	1.9
SO_21982987	3	8385	10	Y	30.8
SO_39656670	3	-	-	N	-
SO_23321300	3	-	-	N	-

Collected 26 Real-World Stack-Overflow Benchmarks

Could solve **17/26 (65%)** Benchmarks

Encouraging Results on Real Stackoverflow Benchmarks

	Depth	Candidates Explored	Sequences Explored	Solved	Time(s)
SO_11881165	1	15	1	Y	0.54
SO_11941492	1	783	8	Y	12.55
SO_13647222	1	5	1	Y	3.32
SO_18172851	1	-	-	N	-
SO_49583055	1	-	-	N	-
SO_49592930	1	2	1	Y	1.1
SO_49572546	1	3	1	Y	1.1
SO_13261175	1	39537	18	Y	300.2
SO_13793321	1	92	1	Y	4.16
SO_14085517	1	10	1	Y	2.24
SO_11418192	2	158	1	Y	0.71
SO_49567723	2	1684022	2	Y	753.1
SO_13261691	2	65	1	Y	2.96
SO_13659881	2	2	1	Y	1.38
SO_13807758	2	711	2	Y	7.21
SO_34365578	2	-	-	N	-
SO_10982266	3	-	-	N	-
SO_11811392	3	-	-	N	-
SO_49581206	3	-	-	N	-
SO_12065885	3	924	1	Y	0.9
SO_13576164	3	22966	5	Y	339.25
SO_14023037	3	-	-	N	-
SO_53762029	3	27	1	Y	1.9
SO_21982987	3	8385	10	Y	30.8
SO_39656670	3	-	-	N	-
SO_23321300	3	-	-	N	-

Collected 26 Real-World Stack-Overflow Benchmarks

Could solve **17/26 (65%)** Benchmarks

Can find programs containing **three**-function sequences

Encouraging Results on Real Stackoverflow Benchmarks

	Depth	Candidates Explored	Sequences Explored	Solved	Time(s)
SO_12065885	3	924	1	Y	0.9
SO_13576164	3	22966	5	Y	339.25
SO_53762029	3	27	1	Y	1.9
SO_21982987	3	8385	10	Y	30.8

90% of Accepted Answers on StackOverflow contain **upto 3 functions**

Collected 26 Real-World Stack-Overflow Benchmarks

Could solve **17/26 (65%)** Benchmarks

Can find programs containing **three**-function sequences

Encouraging Results on Real Stackoverflow Benchmarks

	Depth	Candidates Explored	Sequences Explored	Solved	Time(s)
SO_11881165	1	15	1	Y	0.54
SO_11941492	1	783	8	Y	12.55
SO_13647222	1	5	1	Y	3.32
SO_18172851	1	-	-	N	-
SO_49583055	1	-	-	N	-
SO_49592930	1	2	1	Y	1.1
SO_49572546	1	3	1	Y	1.1
SO_13261175	1	39537	18	Y	300.2
SO_13793321	1	92	1	Y	4.16
SO_14085517	1	10	1	Y	2.24
SO_11418192	2	158	1	Y	0.71
SO_49567723	2	1684022	2	Y	753.1
SO_13261691	2	65	1	Y	2.96
SO_13659881	2	2	1	Y	1.38
SO_13807758	2	711	2	Y	7.21
SO_34365578	2	-	-	N	-
SO_10982266	3	-	-	N	-
SO_11811392	3	-	-	N	-
SO_49581206	3	-	-	N	-
SO_12065885	3	924	1	Y	0.9
SO_13576164	3	22966	5	Y	339.25
SO_14023037	3	-	-	N	-
SO_53762029	3	27	1	Y	1.9
SO_21982987	3	8385	10	Y	30.8
SO_39656670	3	-	-	N	-
SO_23321300	3	-	-	N	-

Collected 26 Real-World Stack-Overflow Benchmarks

Could solve **17/26 (65%)** Benchmarks

Most solutions (**11/17**) found in top-10 function sequences explored

Summary

RISELab goal: Develop open source platforms, tools and algorithms for **r**eal-time **i**ntelligent decisions, decisions that are **s**ecure and **e**xplainable

Exciting research at intersection of systems and ML

New distributed algorithms and systems for ML

Solve system and engineering problems with ML

- Solutions with guaranteed correctness & explainable