Spring25 CS598YP

14.2 Bao: Optimizing *Query-level Hint*

Yongjoo Park

University of Illinois Urbana-Champaign

Most slides from Ryan Marcus' SIGMOD'21 talk: https://www.youtube.com/watch?v=nEy90-WNkjo

High-level Comparison: DB-BERT vs BAO

DB-BERT

- system-level tuning
- accelerates the training by *reading from manual*

BAO

- query-level tuning (system configs remain the same)
- leverages existing query optimizer

Background: Hints for query planning

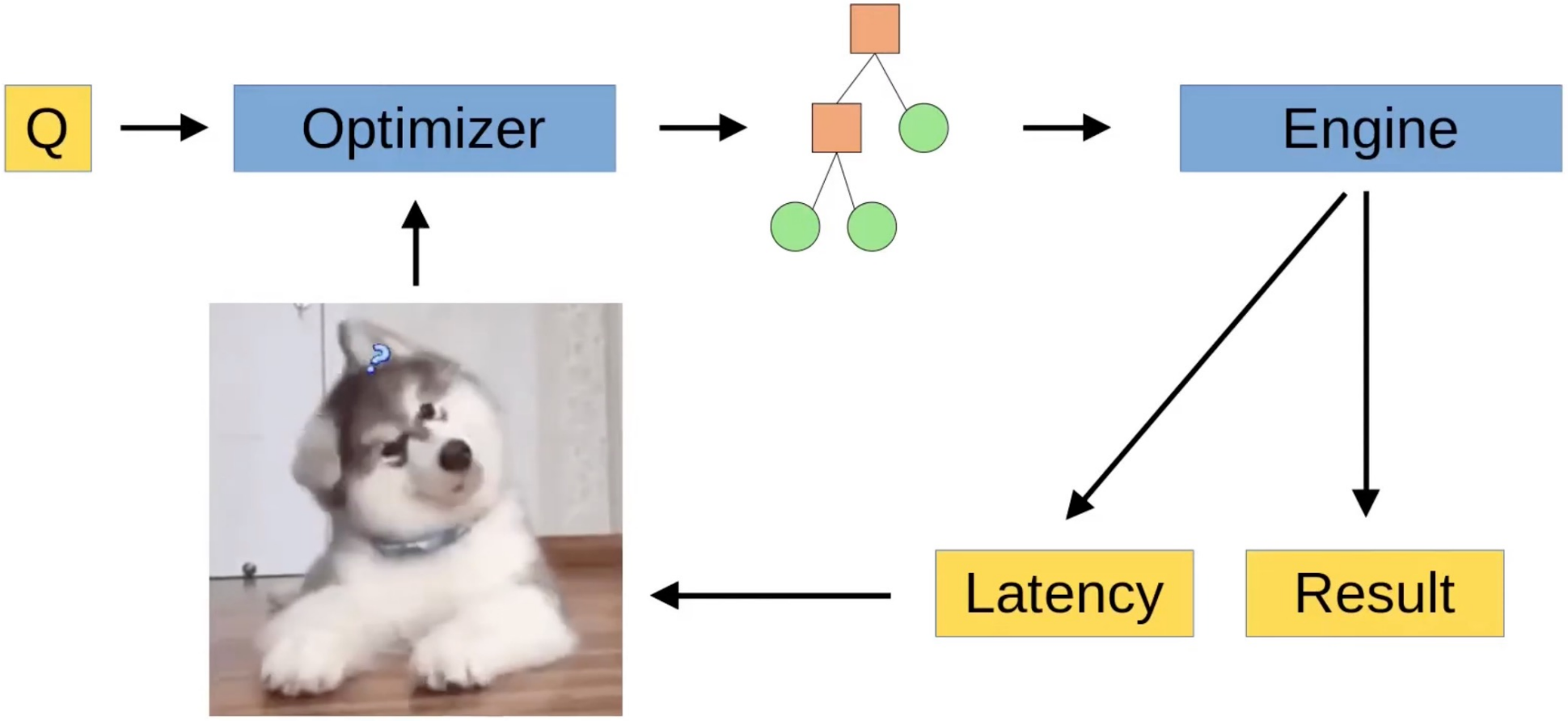
Basic Usage

`pg_hint_plan` reads hinting phrases in a comment of special form given a SQL statement. A hint can be specified by prefixing it with the sequence `"/\*+"` and ending it with `"\*/"`. Hint phrases consist of hint names and parameters enclosed by parentheses and delimited by whitespaces. Hint phrases can use newlines for readability.

In the example below, a hash join is selected as the join method while doing a sequential scan on `pgbench_accounts`:

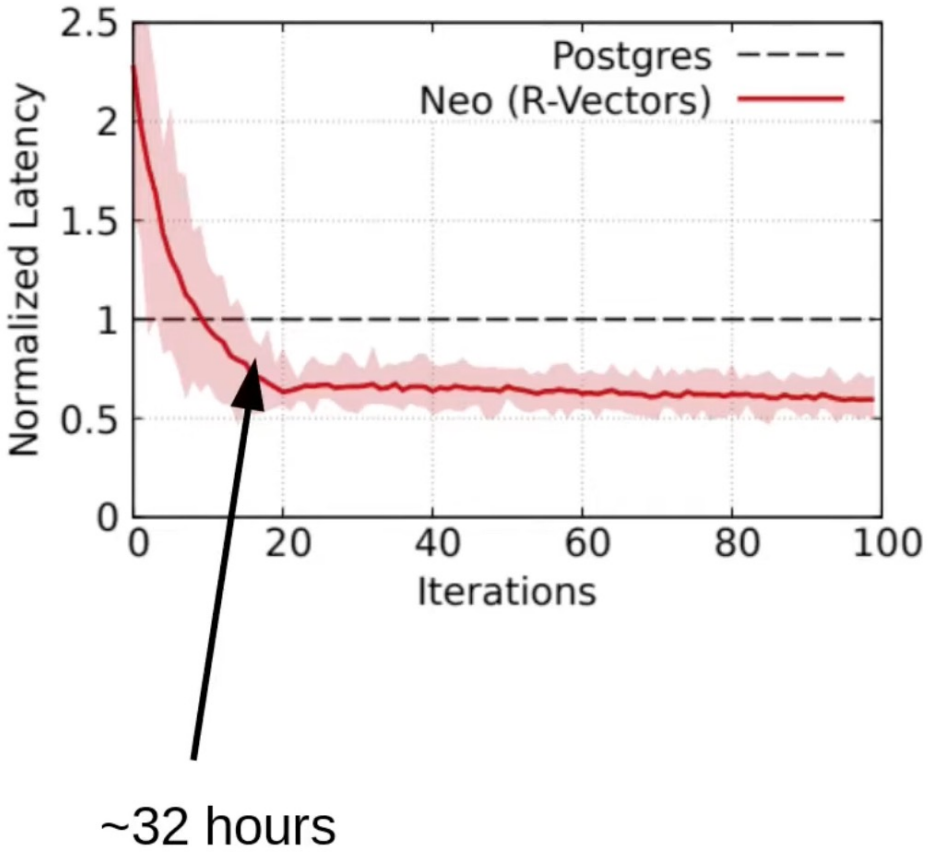
```
=# /*+
     HashJoin(a b)
     SeqScan(a)
    */
   EXPLAIN SELECT *
     FROM pgbench_branches b
     JOIN pgbench_accounts a ON b.bid = a.bid
     ORDER BY a.aid;
                                        QUERY PLAN
-------------------------------------------------------------------------------------
    Sort  (cost=31465.84..31715.84 rows=100000 width=197)
    Sort Key: a.aid
    ->  <b>Hash Join</b>  (cost=1.02..4016.02 rows=100000 width=197)
            Hash Cond: (a.bid = b.bid)
            ->  <b>Seq Scan on pgbench_accounts a</b>  (cost=0.00..2640.00 rows=100000 width=97)
            ->  Hash  (cost=1.01..1.01 rows=1 width=100)
                ->  Seq Scan on pgbench_branches b  (cost=0.00..1.01 rows=1 width=100)
(7 rows)
```

Query optimization workflow



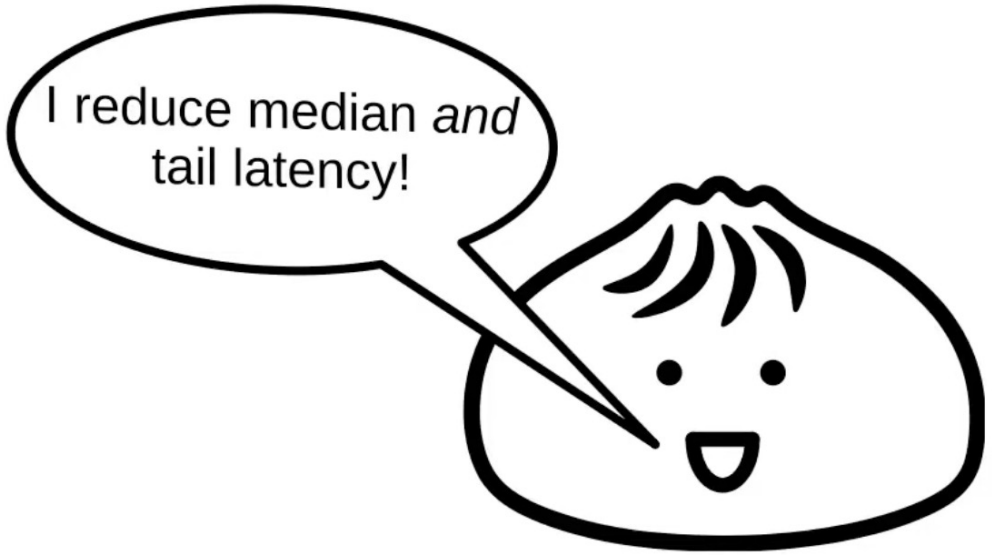
Previous work (Neo) has limitations

- Completely replaced traditional query optimizers.
- Promising results.
- Three main issues
  - Sample (in)efficiency
  - Brittleness
  - Tail catastrophe



Introducing Bao

- Bao: Bandit optimizer
- By *steering* a traditional query optimizer, Bao:
  - Outperforms PG after *1 hour* of training
  - Reduces 99% latency
  - Adapts to changes in workload, schema, and data.

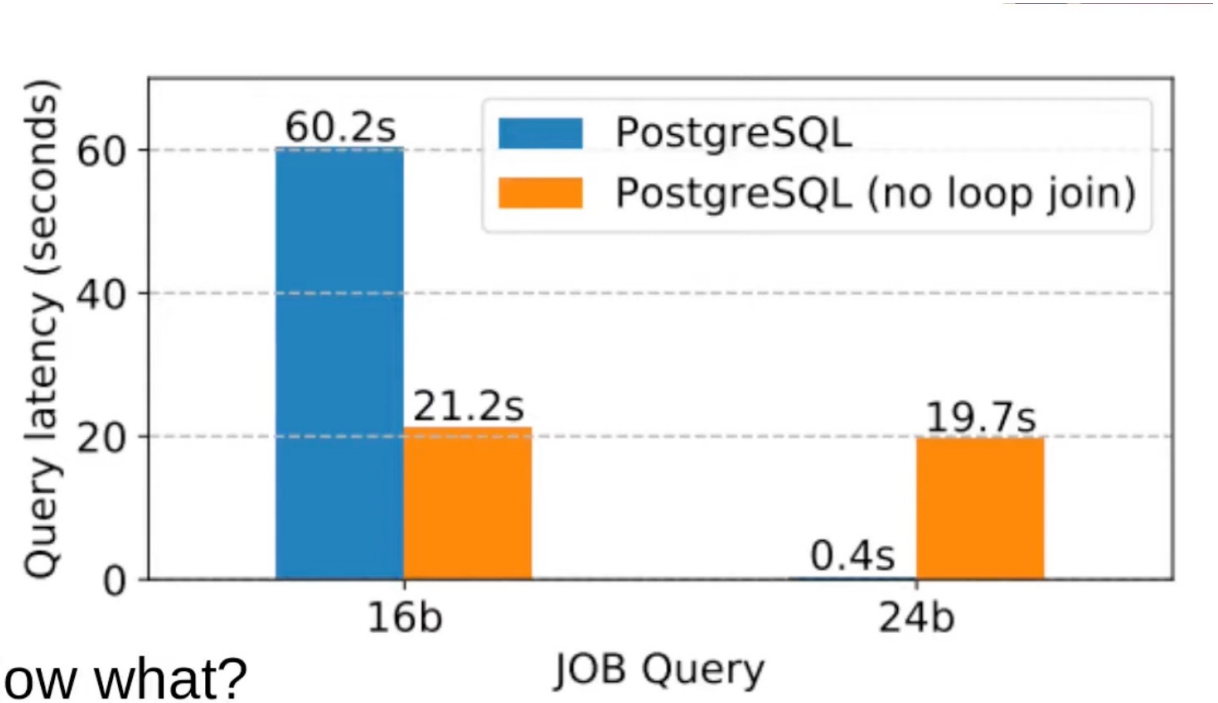


Query hints can be effective

Slow query. Run EXPLAIN.
> Loop join plan,
> Low selectivity

Try disabling loop join
> Huge improvement

Apply this hint globally
> … other regressions



Undo that, need local hints. Now what?

Opt 1: Apply the hint to every instance of the query

Opt 2: Set as default, find regressions, add hints to those queries

Opt 3: Give up

Choosing hints as multi-armed bandit problem

*Given a **query**, we can specify a different **hintset***

Multi-Armed Bandit

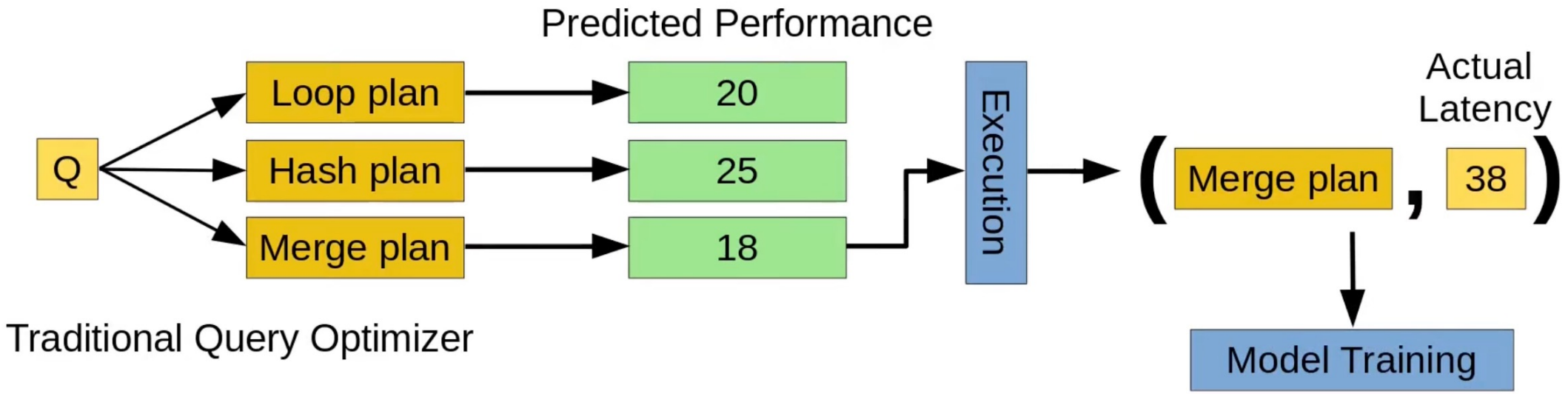
- Each arm is *a hintset*
- Which arm is most rewarding?

TODO: Estimate reward

- Should work across queries
- Should balance exploration and exploitation

Bao workflow: leverages existing optimizer

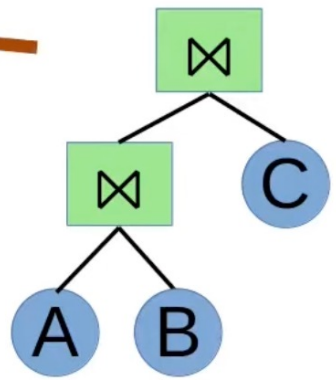
- Bao automatically determines the right hint to use.
- Consider different hints as *arms* in a *contextual multi-armed bandit*



Naïve ML approach may ***NOT*** be effective

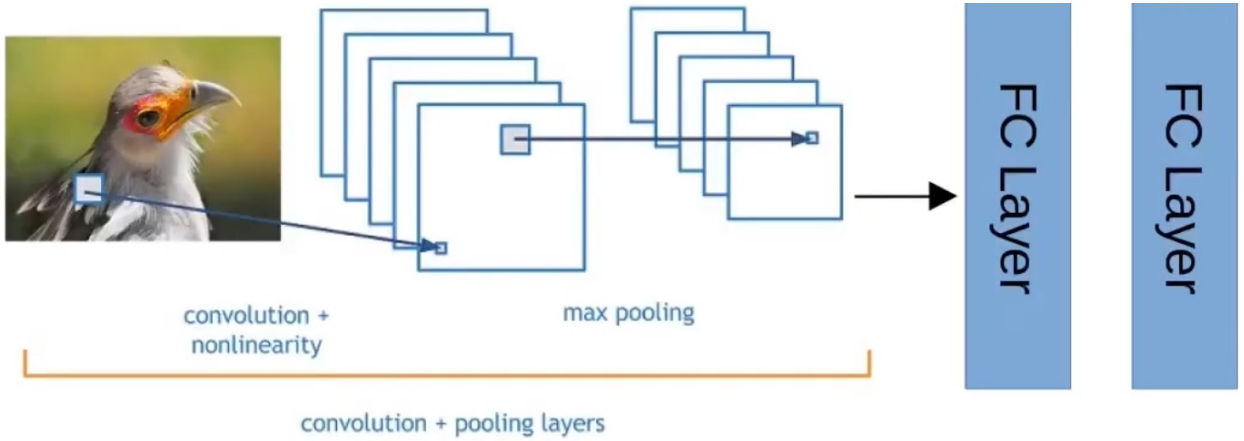
- Bao needs a good predictive model.
- Problem: Query plans have a tree structure.
- ~~Solution: flatten the tree into a vector and engineer some features~~

This is not normally how machine learning is effective.

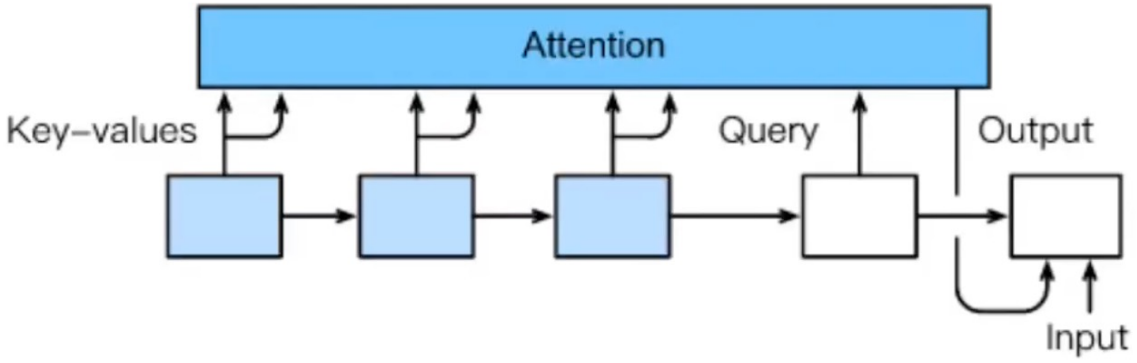


What makes ML good?

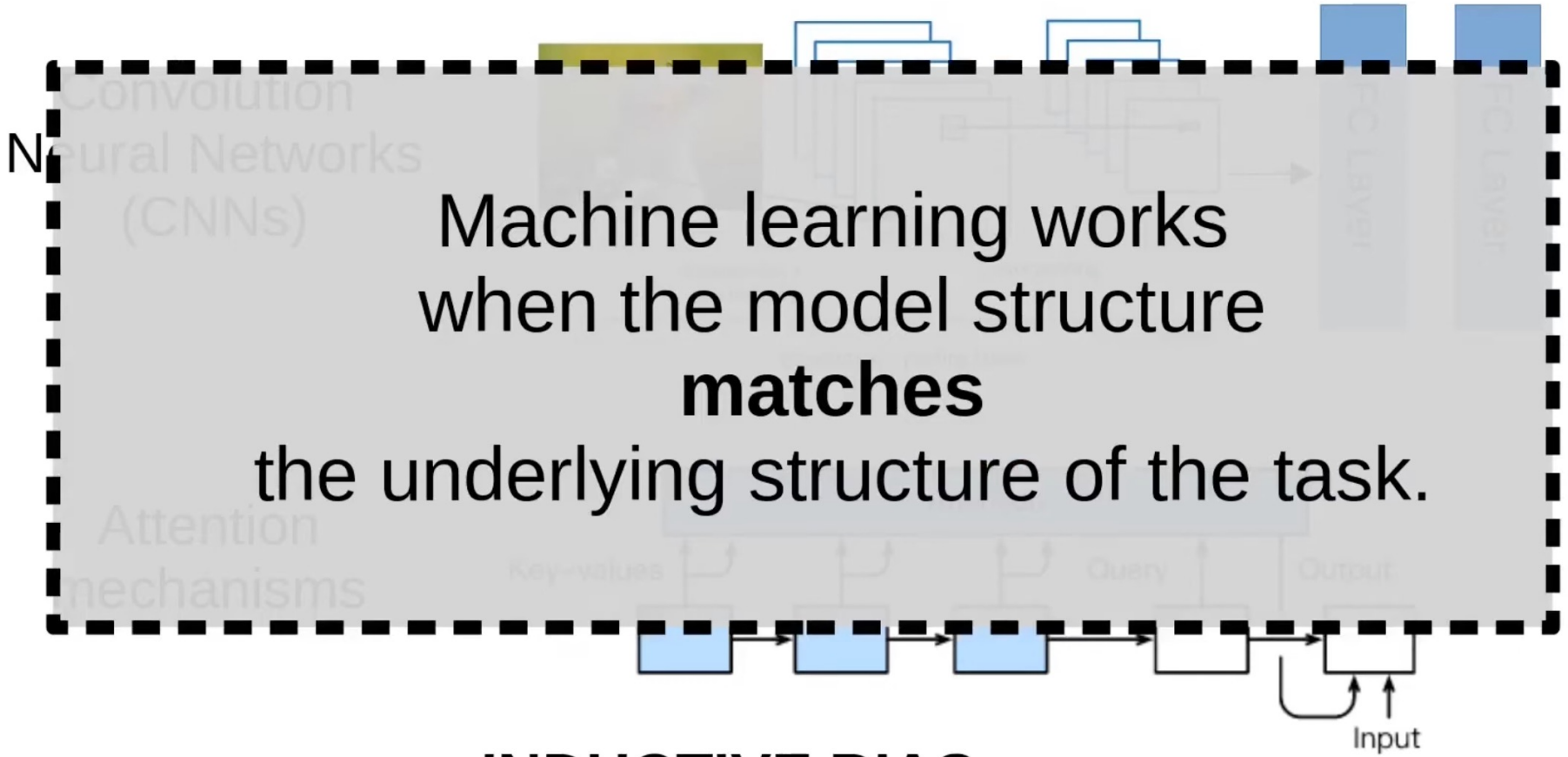
Convolution Neural Networks (CNNs)



Attention mechanisms



What makes ML good?

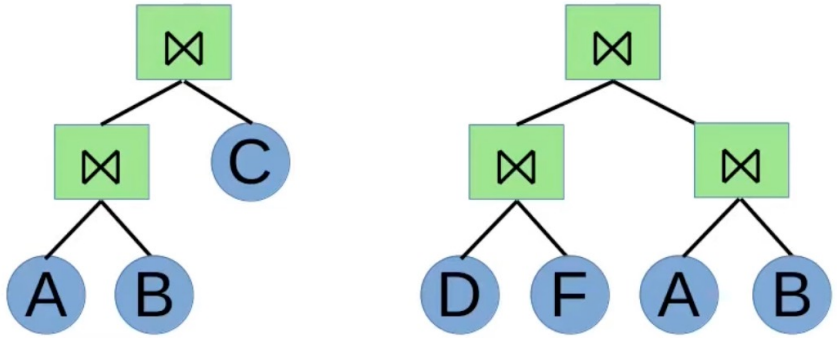


INDUCTIVE BIAS

Adopts tree conventional neural network

erasing
problem structure
and using
fully-connected NNs

integrating
the
structure of the
problem into
the NN itself

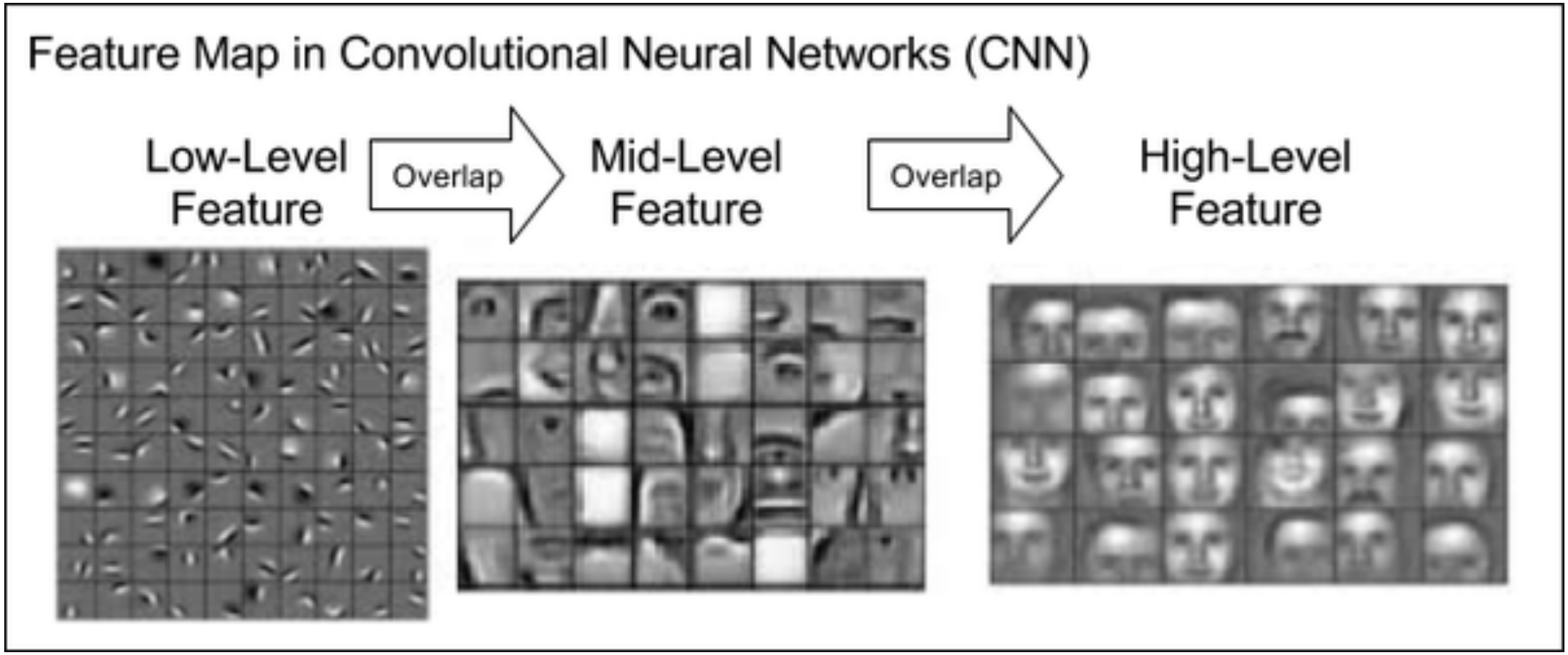


Fundamental structure is a query plan tree.

=> Tree convolution neural networks

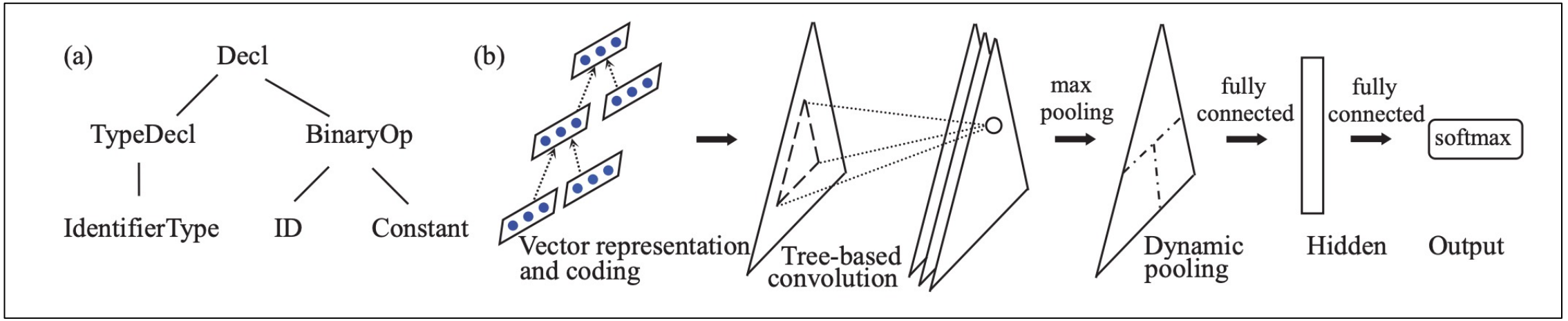
https://ryan.cab/neo
https://ryan.cab/treeconv

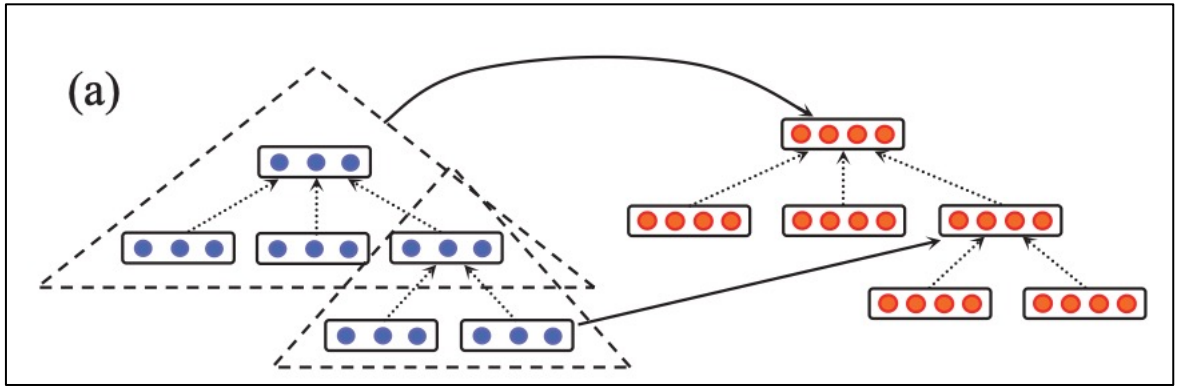
Convolution neural network in computer vision



Tree Convolutional Neural Network (AAAI-16)

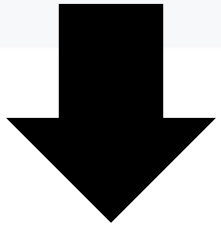
Designed for programming language represented in AST





Query plan is tree-structured

```
   EXPLAIN SELECT *
     FROM pgbench_branches b
     JOIN pgbench_accounts a ON b.bid = a.bid
     ORDER BY a.aid;
                                        QUERY PLAN
-------------------------------------------------------------------------------------
    Sort  (cost=31465.84..31715.84 rows=100000 width=197)
    Sort Key: a.aid
    ->  <b>Hash Join</b>  (cost=1.02..4016.02 rows=100000 width=197)
            Hash Cond: (a.bid = b.bid)
            ->  <b>Seq Scan on pgbench_accounts a</b>  (cost=0.00..2640.00 rows=100000 width=97)
            ->  Hash  (cost=1.01..1.01 rows=1 width=100)
                ->  Seq Scan on pgbench_branches b  (cost=0.00..1.01 rows=1 width=100)
(7 rows)
```



Performance metric (e.g., latency)

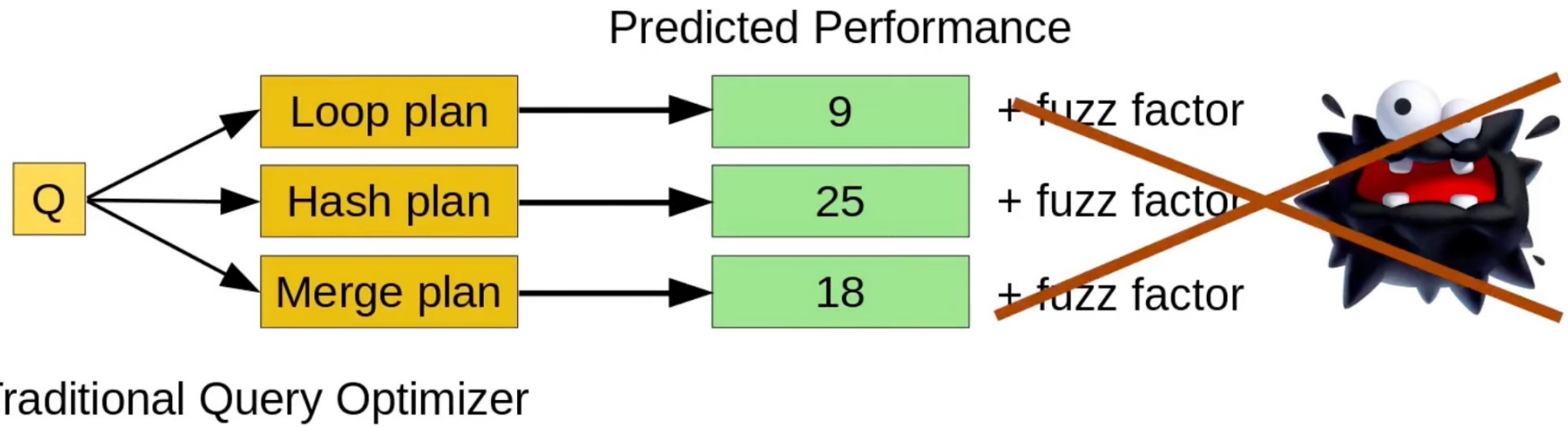
Training

- Option 1: Use a giant query log and train / continuously redeploy the model.
  - “Easy,” but doesn’t adapt.
- Option 2: Periodically retrain the model online using Thompson sampling.

How can we allow ***exploration***?

- How do we balance exploration (new policies) with exploitation (doing what we know works)?

Exploitation: pick the lowest.
Exploration: choose randomly.



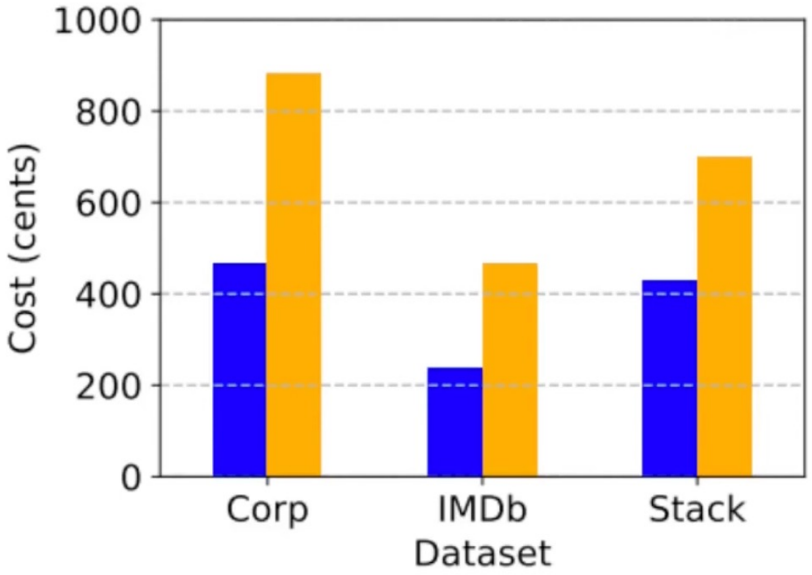
Traditional Query Optimizer

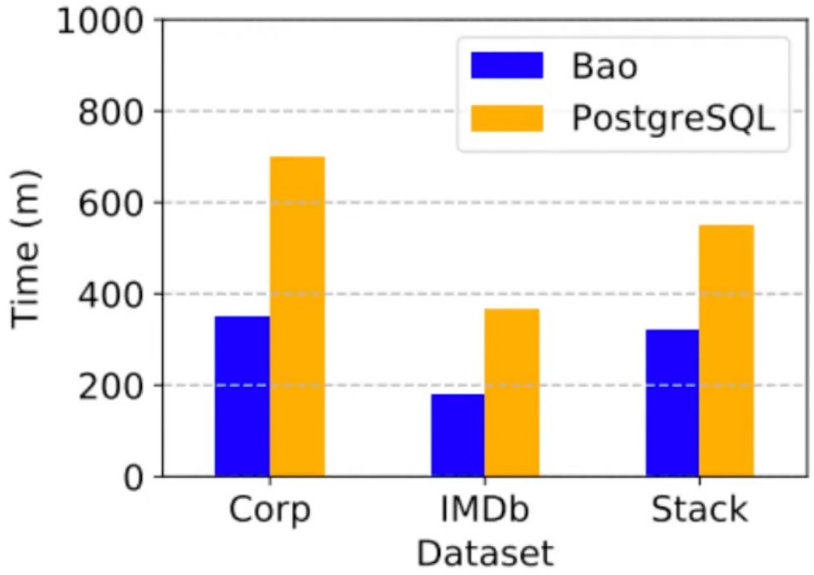
Thompson Sampling

- An old, well-studied algorithm for balancing exploration and exploitation.

Usual ML training (exploitation)	model weights = E[P(model weights | data)]
Pick a random model (exploration)	model weights = sample P(model weights)
Sample model weights (Thompson Sampling)	model weights = sample P(model weights | data)

Bao reduces latency & cost significantly





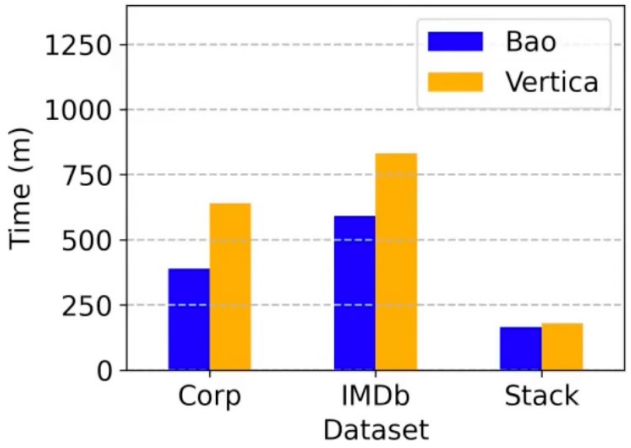
Bao handles dynamic workloads, schema, and data.

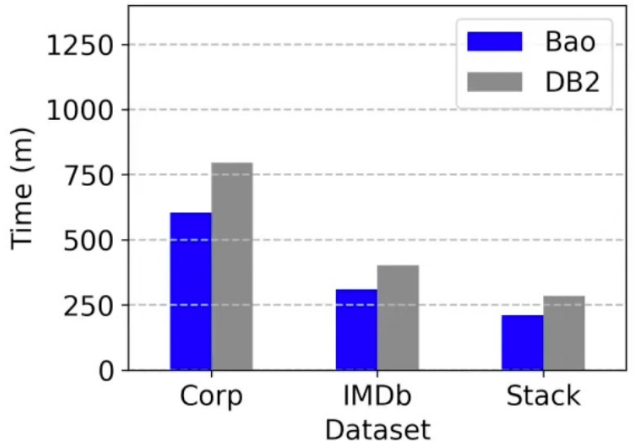
	Size	Queries	WL	Data	Schema
IMDb	7.2 GB	5000	Dynamic	Static	Static
Stack	100 GB	5000	Dynamic	Dynamic	Static
Corp	1 TB	2000	Dynamic	Static[a]	Dynamic

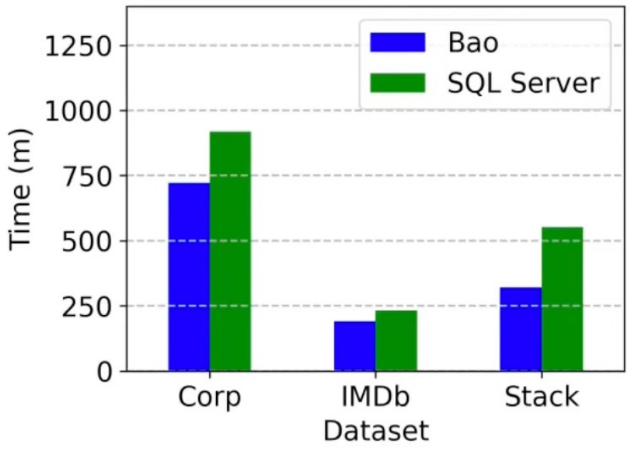
Significant improvement across various systems

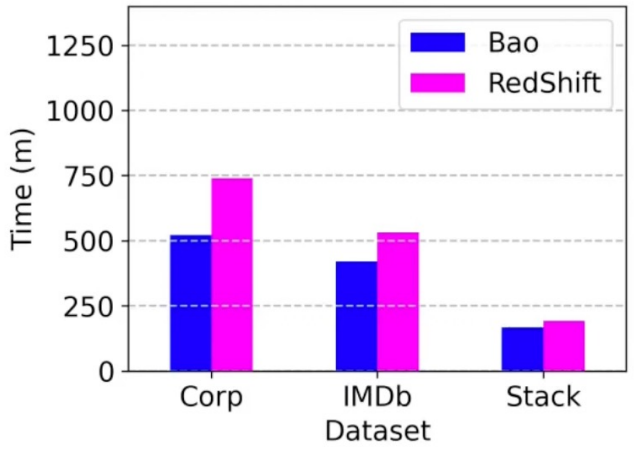
Bao works on commercial and distributed systems as well

(3 node clusters, each ran on their preferred cloud provider with the cheapest nodes available)



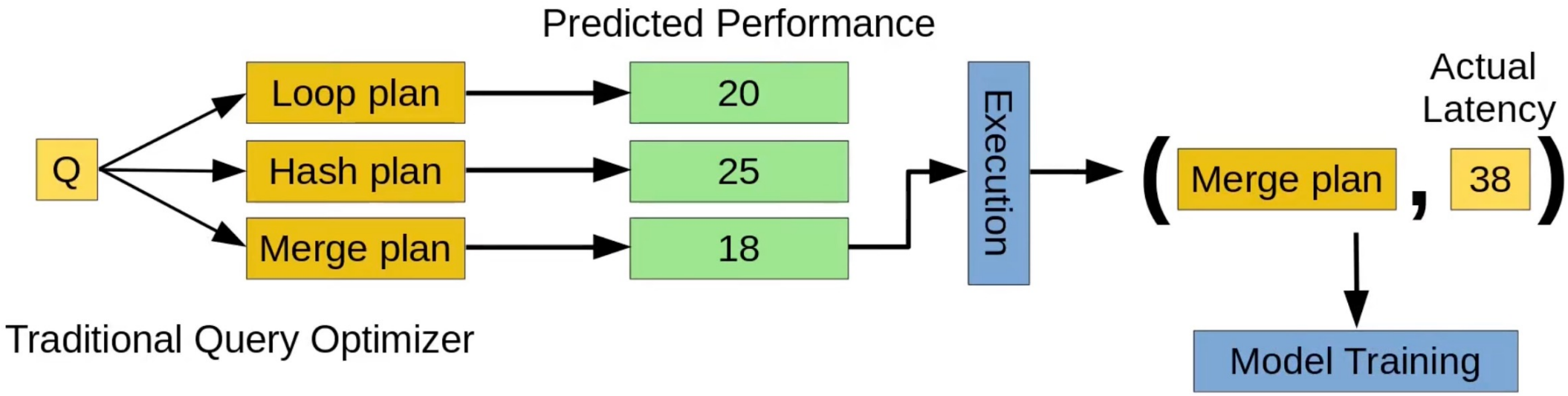






Summary of Bao

- Aims to provide **good** query hints
- Quality of hints are evaluated using **Tree-CNN**
- Balance (*explore vs exploit)* through **Bandit Optimizer**



Questions?