Lecture #02

Carnegie Mellon University

Special Topics:

Self-Driving Database Management Systems

Self-Driving Database Overview

@Andy_Pavlo // 15-799 // Spring 2022

ADMINISTRIVIA

Your first reading review was due before class.

Everyone enrolled should have been invited to the **15-799-spring2022** CMU-DB Slack Channel.

Everyone must sign up for at least one paper presentation date.

Project #1 will be announced on Wed Jan 26th.
→ I will email your EC2 credit code.

ADMINISTRIVIA

Frid Jan 28: I am moving my office hours to 1:30pm for this week only.

Mon Jan 31: First in class lecture (GHC 4303)
→ Please wear a mask the entire time.
→ No food or drink during the lecture.

Mon Jan 31: First speaker at CMU-DB Vaccination Seminar Series (Zoom).

LAST CLASS

We discussed the tasks that we want to automate in a DBMS to achieve autonomous operation.

→ Overview of what changes one can make to a DBMS to tune it for a given application.

→ These are things that human DBAs do now.

DATABASE OPTIMIZATION

Physical Database Design

Knob Configuration

Query Tuning

Hardware Provisioning

TODAY'S AGENDA

Self-Driving Database Definition
Workload Modeling
Behavior Modeling
Action Planning

DATABASE AUTOMATION

People have been trying to automate the management and optimization of DBMS since the first systems were built in the 1970s.

What is different about a self-driving DBMS?

We need to classify levels of automation like the SAE J3016 for autonomous vehicles.

AUTONOMOUS OPERATION LEVELS

Level	*Name*	*Description*
#0	Manual	No autonomy.
#1	Assistant	Recommendations to assist user.
#2	Mixed	Automatically identify when to recommend actions to the user.
#3	Local	Self-contained components can optimize themselves automatically.
#4	Directed	Semi-autonomous with direction from user.
#5	Self-Driving	Fully autonomous without direction.

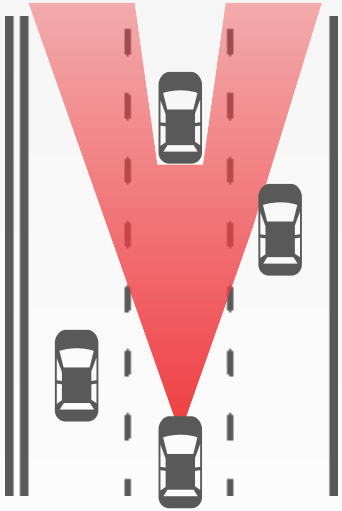
SELF-DRIVING DATABASE

A self-driving DBMS automatically selects and deploys ***actions*** to improve the system's ***objective function*** without any human intervention.

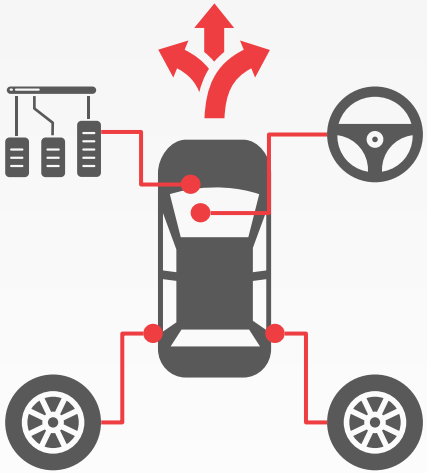
What makes this different than previous methods is that the DBMS must reason about time in the decision making process.

SELF-DRIVING CAR

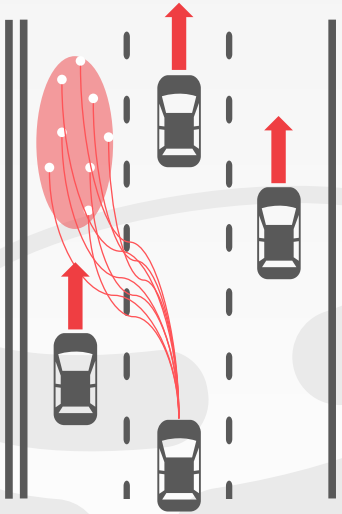
Perception



Action Model



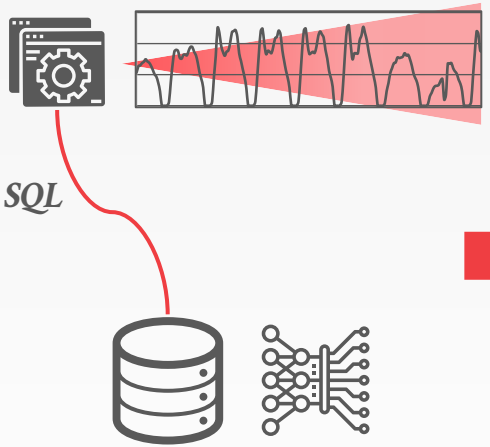
Planning



Source: Lin Ma

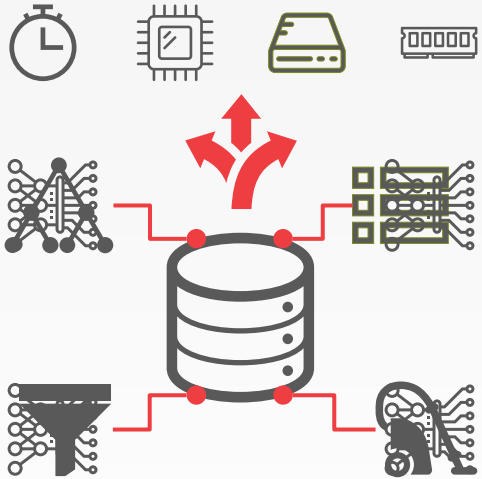
SELF-DRIVING DATABASE

Perception



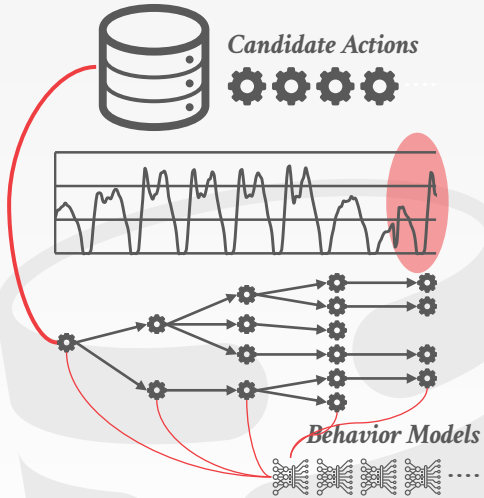
Workload Forecasting

Action Model



Behavior Modeling

Planning



Action Planning

Source: Lin Ma

WORKLOAD FORECASTING

Predict what queries the application will execute in the future to determine what actions to deploy to make sure that it can meet its target objective requirements.

Example: The stock market opens at 9am. We need to make sure that we have the right indexes installed before this deadline.

DATABASE WORKLOADS

Almost no database workloads are completely random. Our automation methods can take advantage of their cyclic and repeating nature.

Workloads dimensions that we can exploit:
→ **Temporality**
→ **Contents**

WORKLOADS: TEMPORALITY

Humans have predictable patterns:
→ Working hours, weekends, holidays

Businesses have predictable patterns:
→ Business hours, weekly/monthly/quarterly reporting

A DBMS's future workload will be similar to some previous period in the past.
→ We will need to account for growth trends.

Caveat: Flash mobs are impossible to predict.

WORKLOADS: CONTENTS

OLTP Workloads:

→ Applications repeatedly execute the code that invoke the same queries (same template, different params).

OLAP Workloads:

→ Queries for dashboards or reporting have the same shape but with different computations (filters, aggregates).

The types of queries that the database will execute in the future will be similar to what it has executed in the past.

Caveat #1: New features create new queries.

Caveat #2: Ad-hoc maintenance queries.

WHY THIS MATTERS

If we have an accurate workload forecast, this allows the DBMS to discern many things:

→ What actions could provide the most benefit.
→ How the database's contents will evolve over time.
→ When are peak times during the day/week/month.

BEHAVIOR MODELING

Understand and measure the DBMS's behavior for a given ***state*** when executing a workload.

→ Input: Database State + Workload

→ Output: Runtime Operations

The output could either be the target objective or some proxy measurement that we then must map to the objective.

DATABASE STATE

A DBMS state represents its condition with respect to the following aspects of its existence:

→ Database Contents (User Data + Catalog + Statistics)
→ Physical Design
→ Knob Configuration
→ Hardware Configuration

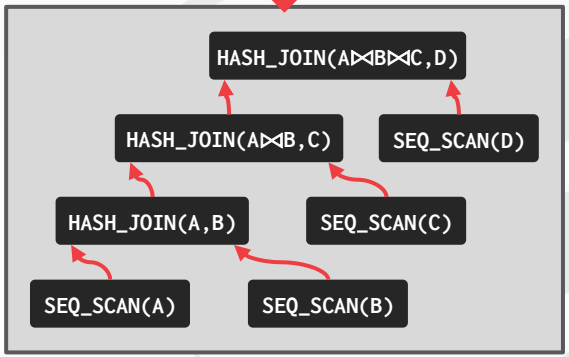
BEHAVIOR MODELING: QUERY PLANS

The most common way to model a DBMS's behavior is through query plans. But this may not provide enough information about the system's runtime behavior.

Fails to capture other aspects of the DBMS's runtime operations.

Plan costs are also internal metrics.

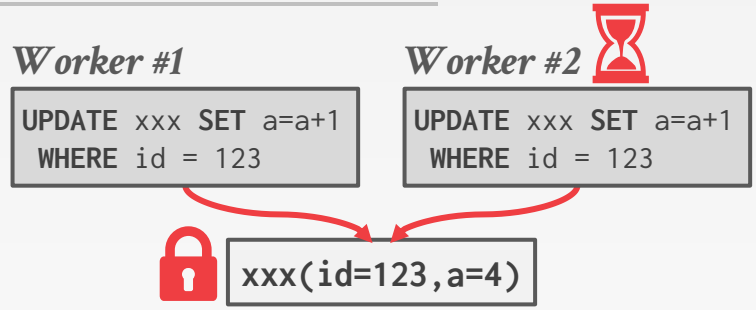
```
SELECT * FROM A
  JOIN B ON A.id = B.id
  JOIN C ON A.id = C.id
  JOIN D ON A.id = D.id
 WHERE B.val = 'WuTang'
   AND D.val = 'Clan';
```



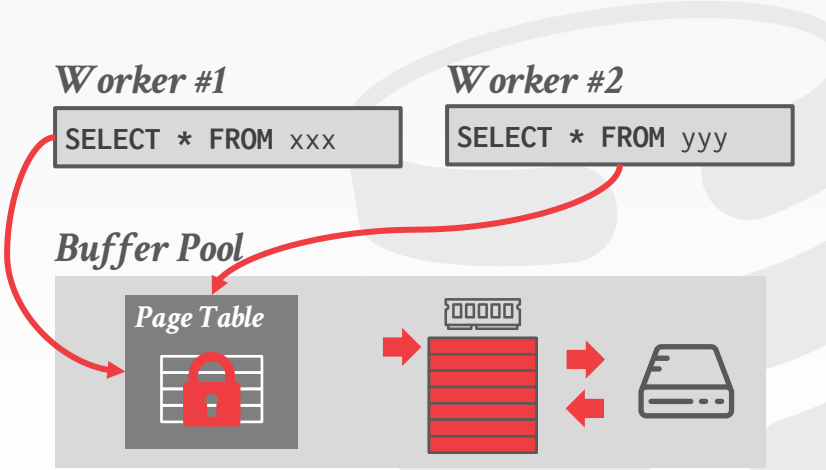
Estimated Cost: 1000

BEHAVIOR MODELING: CONTENTION

Logical: Queries accessing or modifying the same data at the same time.



Physical: Queries interfering with the execution of other queries or maintenance operations.



TRAINING DATA COLLECTION

We need a principled way to generate the data that we need to construct the models to predict the behavior of the DBMS for new actions.

We will want to vary the state and workload.

The effects of these changes are not consistent:

→ Example: Doubling the # of CPU cores does not magically double the DBMS's performance.

WHY THIS MATTERS

When behavior models are combined with forecasting models, we can now predict:

→ How much hardware resources will the system need.

→ Whether the database is moving towards a critical state.

ACTION PLANNING

The goal of this step is to identify what action to apply at a given time (now or in the future) and determine how to apply it.

There are two parts of this problem:
→ **Action Enumeration**
→ **Action Selection**

ACTION ENUMERATION

For a given database state, generate the set of actions that the DBMS could deploy using heuristics.

→ The goal is to cut down the number of actions the system considers using a more expensive cost model.

→ Some actions always exist (e.g., knobs, hardware scaling).

Example: Generate candidate indexes by extracting columns that are referenced in **WHERE** clauses.

ACTION ENUMERATION

We can exploit our knowledge of databases to reduce the complexity of the problem.

Example #1: Skip single-key indexes on Boolean columns.

Example #2: Skip hash indexes on columns accessed with only non-equality predicates.

ACTION ENUMERATION

We need to be careful that we do not introduce assumptions that are not always true in the enumeration process.

Example: Whether an index on a set of columns is as good candidate action depends on the number of tuples, distribution of values, and workload access patterns.

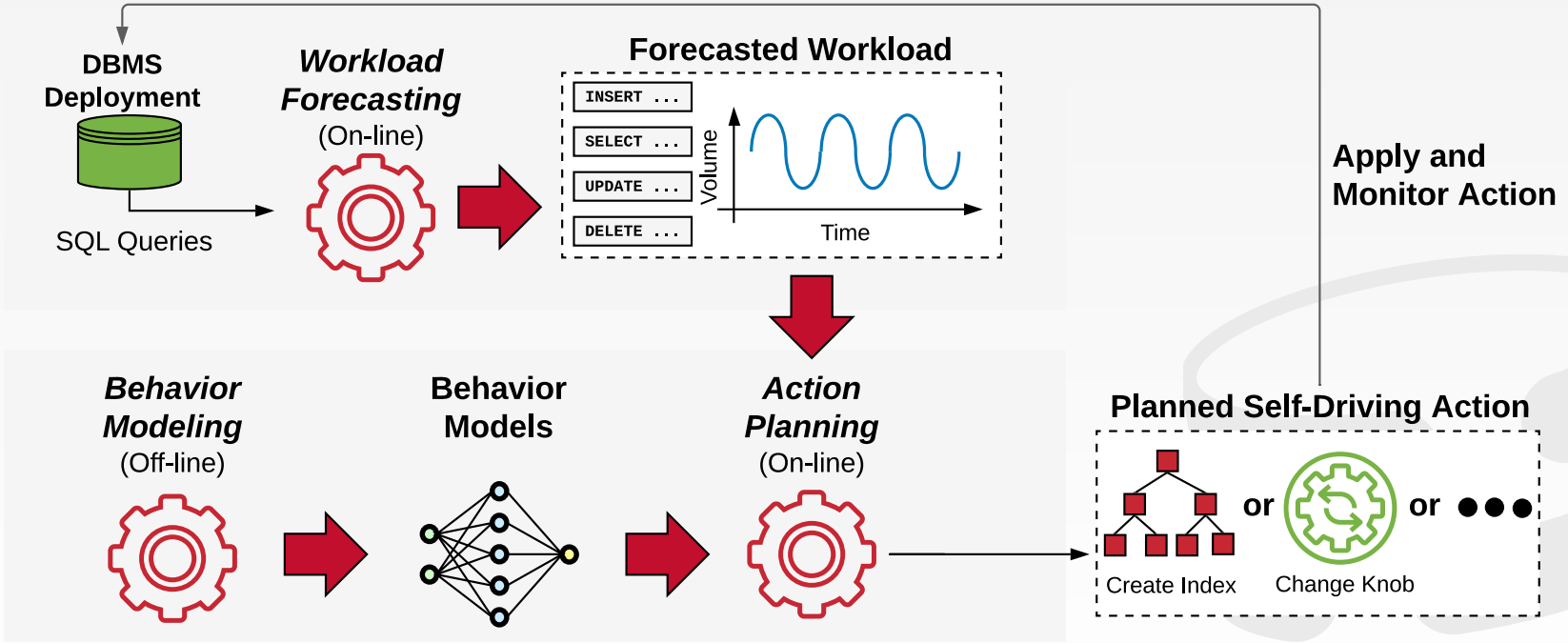
ACTION SELECTION

For the set of candidate actions, generate a sequence of actions that the system identifies as providing the most benefit to its target objective.

There are many different ways to do this that we will cover this semester.

→ Tree search
→ Reinforcement Learning

END-TO-END SYSTEM



Source: Lin Ma

PARTING THOUGHTS

DBMSs are complex with many moving parts.

But unlike with self-driving cars, we do **not** need to be perfectly accurate in our decisions.

For the next four weeks, we will study methods for behavior modeling, action enumeration, and action selection.

→ We will assume that the workload is static.

NEXT CLASS

Index Recommendation

Project #1