Special Topics:
Self-Driving Database Management Systems

Behavior Modeling I
We discussed Utah's query plan encoder paper. → It sort of was a workload / behavior modeling paper.

But it can only predict the behavior of queries based on high-level measurements (e.g., latency).
TODAY’S AGENDA

Behavior Modeling
Project #2 Topics
Generate *behavior models* that predict the cost and benefit for self-driving actions.
Create an index with different number of threads.
CHALLENGES

High-dimensionality
Concurrent operations
Data collection and training
Interpretability, debuggability, and adaptivity
PREVIOUS SOLUTIONS

Analytical Models
→ Onerous design from experts
→ Difficult to adapt

ML Models
→ Focus on single query execution
→ Difficult to generalize across databases
OFF-LINE BEHAVIOR MODELING

ModelBot 2 (MB2)

Specialized Runners ➔ DBMS ➔ Training Framework ➔ Behavior Models

- Lightweight Metrics System
- Operating Unit Models
- Interference Model
## OPERATING UNIT MODELS

Decompose the DBMS into small operating units (OUs) to model separately.

<table>
<thead>
<tr>
<th>OU Examples</th>
<th>Input Features</th>
<th>Output Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Hash Table</td>
<td><img src="image1" alt="Knobs" /></td>
<td><img src="image2" alt="Elapsed time" /></td>
</tr>
<tr>
<td>Create Index</td>
<td><img src="image3" alt="Knobs" /></td>
<td><img src="image4" alt="CPU Metrics" /></td>
</tr>
<tr>
<td>Serialize Log Records</td>
<td><img src="image5" alt="Knobs" /></td>
<td><img src="image6" alt="I/O Metrics" /></td>
</tr>
</tbody>
</table>

Source: Lin Ma
OU CATEGORIES

DBMS

- Singular OUs
  - Build Hash Table
  - (1) # rows
  - (2) # columns
  - (3) column size
  - (4) estimated cardinality
  - (5) related knobs

- Batching OUs
  - Serialize Log Records

- Contending OUs
  - Parallel Create Index
  - Must be able to derive all input features without executing query.
Specialized runners exercise each OU through SQL-based synthetic benchmarks.
OU BOUNDARIES

Setup Phase

Execution Engine
- CPU Probe
- Memory Probe
- Disk Probe
- Network Probe

Kernel-level Probes

TS Framework

BPF Program

Codegen

BPF Compiler (BCC)

User-level Probes

Database Management Systems

Query Plan

OU Code

BPF Compiler (GCC, Clang)

C++ Compiler

BEGIN
For tuple in table:
... END
FEATURES

SELECT * FROM foo WHERE ...

DBMS

Processor

Filter

Archive

User-Space

Kernel-Space

Collector

Buffers

BPF Maps

Kernel-level Probes

BPF Program
Training data for OUs may be expensive to collect.

Output Labels

Build Hash Table
Sort
Create Index

Complexity

$n$
$n \log n$
$n \log n$

“Per-tuple” Output Labels

Usually converge within 1M rows
SINGLE-THREAD GENERALIZATION

Evaluate the accuracy of MB2's models for both OLAP and OLTP workloads:
→ Generalize to different scale factors or datasets

MB2 always uses the same OU models generated using off-line training data.

Comparison: QPPNet
→ Query plan-level modeling (Marcus et al., 2019)
→ Train on one workload, evaluate on others (without data generation mechanism)
QUERY LATENCY PREDICTION

OLAP

<table>
<thead>
<tr>
<th>Dataset</th>
<th>QPPNet</th>
<th>MB2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-H 0.1G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPC-H 1.0G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPC-H 10G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Avg. Relative Error

OLTP

<table>
<thead>
<tr>
<th>Dataset</th>
<th>QPPNet</th>
<th>MB2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmallBank</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TATP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Avg. Absolute Error (us)

Source: Lin Ma
SINGLE-THREADED EVALUATION

Simulate a daily transactional-analytical cycle
→ Alternate between TPC-C and TPC-H
→ “Oracle” changes a knob for TPC-H and builds an index for TPC-C

![Actual Query Latency vs Estimated Query Latency](chart.png)

- Normalized Avg. Query Latency
- Actual Query Latency
- Estimated Query Latency

Source: Lin Ma
INTERFERENCE MODEL

Use resource metrics from the OU-model outputs to approximate the interference.

→ **Input Features:** summary statistics of the OU-model outputs for concurrent OUs in a forecasting interval.

→ **Output Labels:** adjustment ratio based on the interference in system when executing workload.

Concurrent runners
INFERENCE PROCEDURE

Concurrent OUs

OU Models

OU Model Outputs

Summary Statistics of the Resource Competition

Interference Model (shared)

Interference Model Outputs
MULTI-THREADED EVALUATION

Start with TPC-C workload.
Switch to TPC-H workload after 25 seconds.
FUTURE WORK

May need to also instrument DBMS background tasks with OUs.

Automatically determine what input features to include per OU.

Automatically devise methods for collecting training data
→ Synthesize runners
→ Sweep workloads
→ Execute workload traces
PARTING THOUGHTS

Behavior modeling is a fundamental step towards building self-driving databases.

A decomposed behavior modeling framework with OU models and runners.
PROJECT #2

Each group (3-4 people) will choose a project that satisfies the following criteria:
→ Relevant to the materials discussed in class.
→ Requires a significant programming effort from all team members.
→ Unique (i.e., two groups cannot pick same idea).
→ Approved by me.

All projects will be based on the NoisePage Pilot with Postgres.
PROJECT #2

Project deliverables:
→ Proposal
→ Status Update
→ Design Document
→ Code Review
→ Final Presentation
→ Code Drop

Project #3
PROJ ECT #2 – PROPOSAL

10-minute presentation to the class that discusses the high-level topic.
→ **Date:** Monday March 14th

Each proposal must discuss:
→ Overall architecture and implementation timeline.
→ How you will test whether your implementation is correct.
→ What workloads you will use for your project.
10-minute presentation to update the class about the current status of your project.

→ Date: Monday April 11th

Each presentation should include:
→ Current development status.
→ Whether your plan has changed and why.
→ Anything that surprised you during coding.
As part of the status update, you must provide a design document that describes your project implementation:

→ Architectural Design
→ Design Rationale
→ Testing Plan
→ Trade-offs and Potential Problems
→ Future Work
PROJECT #2 – CODE REVIEW

Each group will be paired with another group and provide feedback on their code.

Grading will be based on participation.
PROJECT #2 – FINAL PRESENTATION

10-minute presentation on the final status of your project during the scheduled final exam.

You should include any performance measurements or benchmarking numbers for your implementation.

Demos are always hot too…
A project is **not** considered complete until:

→ The code can merge into the main branch without any conflicts.
→ All comments from code review are addressed.
→ The project includes test cases that correctly verify that implementation is correct.
→ Source code contains clear documentation / comments.

Project #3 will be a class effort to merge all the projects together.
PROJECT TOPICS

Expanded OU Coverage
Async OU Features
Action Enumeration

Pilot Control Plane
Pilot Catalog
Workload Forecasting++
Metric Forecasting
EXPANDED OU COVERAGE

NoisePage's OU instrumentation currently only supports a subset of query plan operators. We want to also support OLAP workloads.

**Project:** Add additional markers to collect OU training data for all workloads in BenchBase.

→ William has made it easier to identify which queries are missing OU coverage.
ASYNC OU FEATURES

We collect OU input features immediately when encountering markers. But we may be able to delay collecting some features (e.g., query plan meta-data).

**Project:** Implement asynchronous feature collection for queries.
- Coalesce features offline in training service
- Must work with TScout + Hutch
ACTION ENUMERATION

We want a principled and maintainable way of defining rules to generate candidate actions without a cost model to evaluate in search process. → Not sure whether actions can also be ranked.

**Project:** Create a framework that emits actions. 
→ Also maintain history of actions applied. 
→ Talk with Mike about this.
PILOT CONTROL PLANE

Build communication channels between Pilot and other services (DBMS instances, ML components).
→ Pilot initiates internal commands.
→ Pilot maintains state about what commands it has sent to other nodes and stores their responses.

Project: Build it 😊
NoisePage generates lots of training data that are just written out files on local disk. We need a programmatic way on how services can track this data.

**Project:** Create the Pilot's internal catalog

→ You will want to work with the control plane group to decide the interfaces for storing / retrieving data files.
WORKLOAD FORECASTING++

Our re-implementation of QB5000 is limited.

Project:
We need to predict additional information about the database state beyond expected workload.

- Table growth
- Internal metrics (e.g., dead tuples)

**Project:** Create a new service for collecting additional metrics from DBMS and then train forecast models for them.

- We are considering switching to Stanford's [NeuralProphet](https://neuralprophet.readthedocs.io).
PROJECT #2 – NEXT STEPS

Select your team members
Discuss project topics
Meet with Andy (March 7 – 13)
Proposal Presentation (March 14)

I will update the course spreadsheet with dates/times to sign up for meetings.
NEXT CLASS

QPPNet Behavior Modeling