Tiresias: Enabling Predictive Autonomous Storage and Indexing

Michael Abebe, Horatiu Lazu, Khuzaima Daudjee

tiny.cc/tiresias

UNIVERSITY OF WATERLOO

mtabebe@uwaterloo.ca

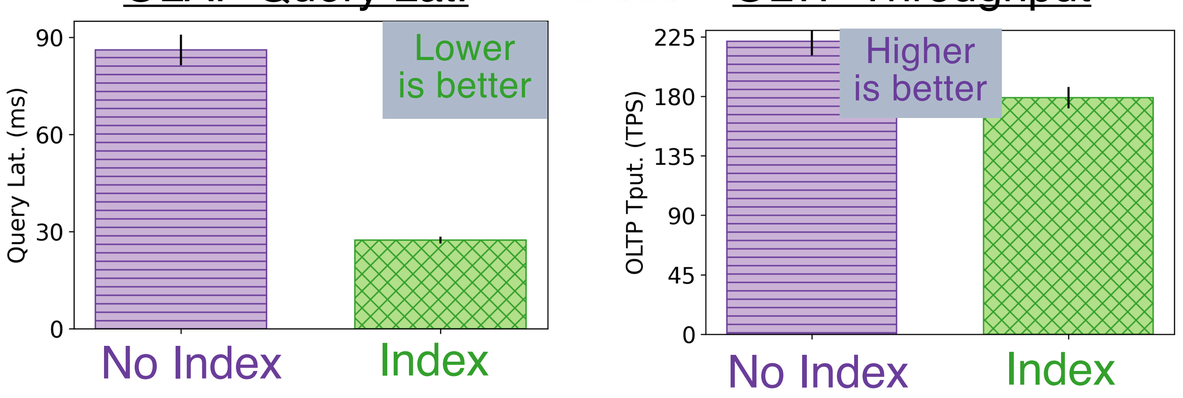
cs.uwaterloo.ca/~mtabebe

github.com/mtabebe

Objective

DBMS **storage** and **indexing choices** have **trade-offs** based on the workload. **Tiresias** enables **adapting** these choices **automatically** based on the **workload**.

OLAP Query Lat. versus OLTP Throughput



Neither indexing choice optimal for both OLTP & OLAP

Architecture

Predict upcoming accesses and latency under different storage and indexing choices

Tiresias

Model workload *access* history

Model access *latencies*

DBMS

Store and index data

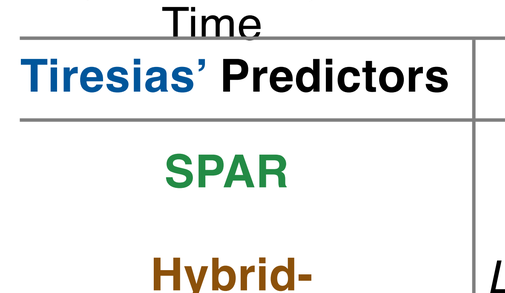
Change data storage and indexing

Report query latency and access history

Tiresias' use cases:

PostgreSQL: automatically add/remove secondary indexes
OLAP DBMS: predictively cracking (sorting) of data
Proteus: Adaptation of storage formats for HTAP

Predicting Upcoming Accesses



User behavior (*e.g., follow the sun*) results in access **patterns**.

Tiresias strives to **learn** and **predict** these access patterns.

Daily and **Weekly** patterns

Tiresias' Predictors	Periodicity	Trend
SPAR	Avg. over *user defined intervals*	Avg. over interval
Hybrid-Ensemble	*Learned* Recurrent Neural Network	Linear Regression

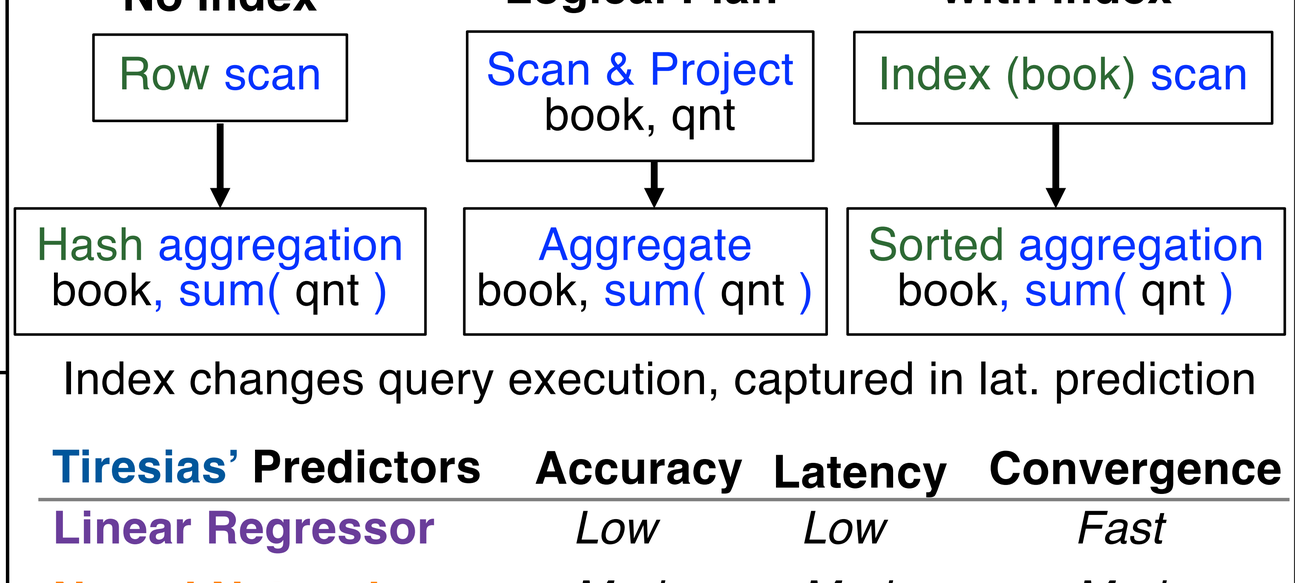
Predicting Transaction Latency

Transactions composed of **physical operators**.

Tiresias **learns** the **latency** of operators under storage/ indexing choices **parameterized** by **workload** statistics.

Tiresias **combines** predicted operators lat. to **predict** txn. lat.

SELECT book, SUM(qnt) GROUP BY book



Index changes query execution, captured in lat. prediction

Tiresias' Predictors	Accuracy	Latency	Convergence
Linear Regressor	*Low*	*Low*	*Fast*
Neural Network	*Med.*	*Med.*	*Med.*
Non-Linear Regressor	*High*	*High*	*Slow*

Making Change Decisions

For Tiresias to make a storage or indexing change:

Expected Benefit > **Upfront Costs**

Predict transaction latency under **current** and **proposed** choice

E.g. with index : predict OLTP txn. will execute 25 ms faster.

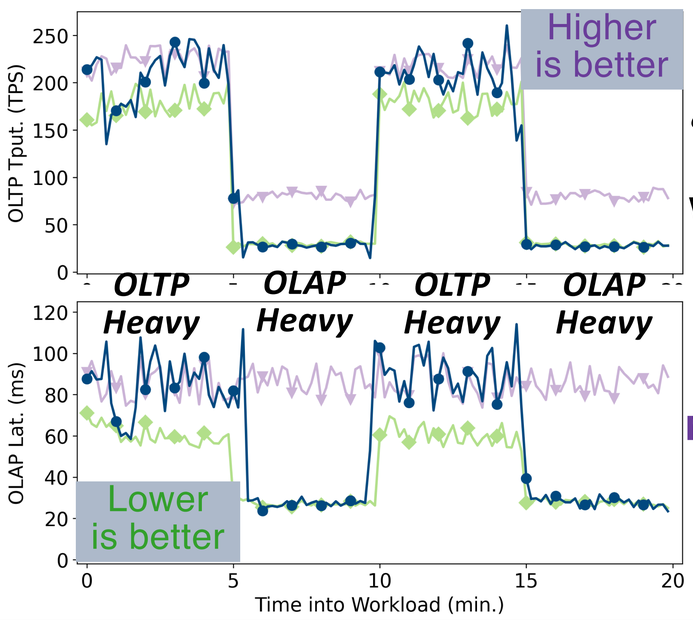
Estimated latency of performing the change

E.g. adding an index will take 300 ms.

Weighted by predicted **likelihood** of transaction occurring in near future, to **balance trade-offs**.

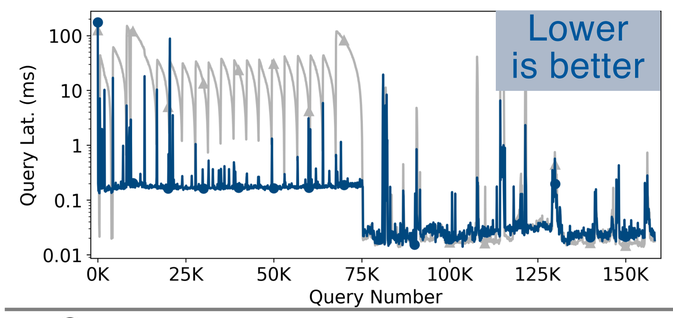
Ensures system does not constantly undo/ redo changes

Experimental Evaluation

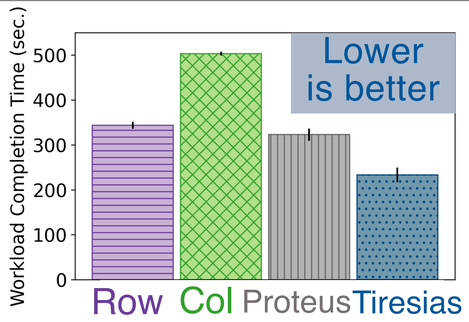


Tiresias automatically *adds* and *removes* **indexes in** *PostgreSQL* in a workload with a *shifting* mix

Tiresias completes workload **faster** than **no indexes** (best for OLTP) & **with indexes** (best for OLAP).



Tiresias enables **predictive cracking**; **reduces** query latency by **40x** versus default cracking in an *OLAP DBMS.*



Tiresias allows **predictive** *storage format adaptation* in **Proteus**

Completes mixed workload (CH) **faster** than *static layouts*: **row** (best for OLTP) & **columnar** (best for OLAP)