Ask Your Doctor Whether Upgrading is Right for You

Automated Upgrade Recommendation for Self-Driving Database Management Systems

Wan Shen Lim, Ruiqi Wang, William Zhang
Jignesh Patel, Andrew Pavlo

Carnegie Mellon University
wanshenl@cs.cmu.edu

Carnegie Mellon Database Group

IAP 2025

Database Upgrades

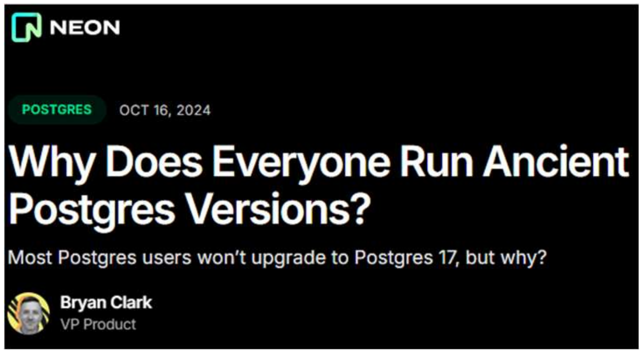
Expectation

People want better DBMSs, right?

PostgreSQL

- Faster aggregations
- Parallel index scan } pg10 (2017)
- Parallel index build } pg11 (2018)
- B-tree deduplication } pg13 (2020)
- Faster parallel seqscans } pg14 (2021)
- Faster sorts } pg15 (2022)
- Efficient garbage collection } pg17 (2024)

Reality



NEON

POSTGRES OCT 16, 2024

Why Does Everyone Run Ancient Postgres Versions?

Most Postgres users won't upgrade to Postgres 17, but why?

Bryan Clark
VP Product

pg9.6: released 2016, end-of-life in 2021

- AWS RDS support until ~~Jan 2022~~ Apr 2022
- Google support until ~~2022~~ 2028?

The Painful Upgrade Process

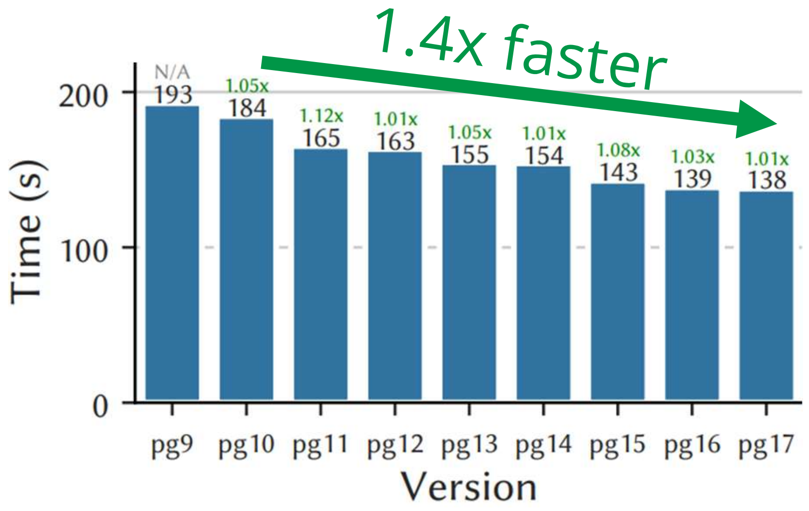
Upgrade Manual → Upgrade Checklist + Rollback Checklist → Mitigate Regressions

Existing upgrade processes focus on **post-upgrade** mitigations

Software Upgrades

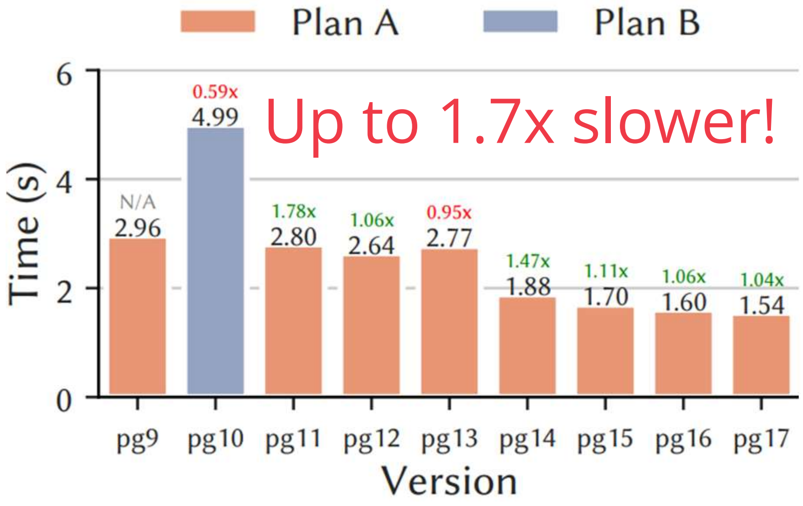
- Executing TPC-H across software versions

All Queries



1.4x faster

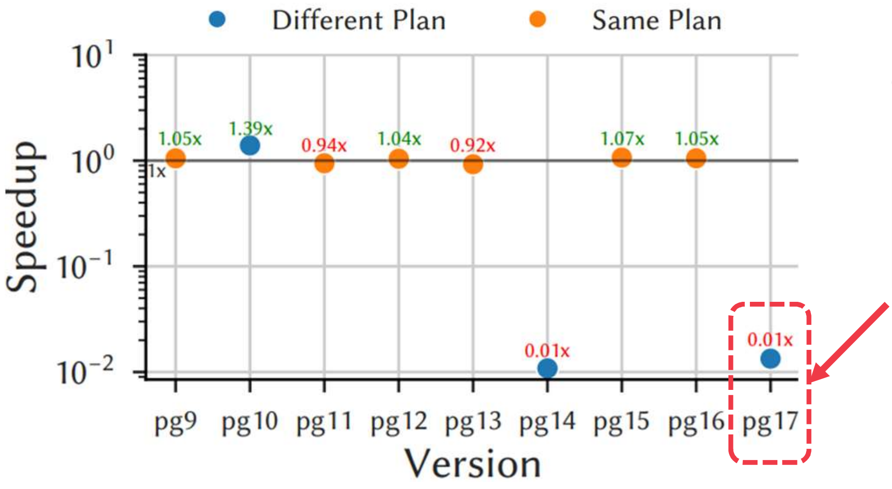
Only Q6



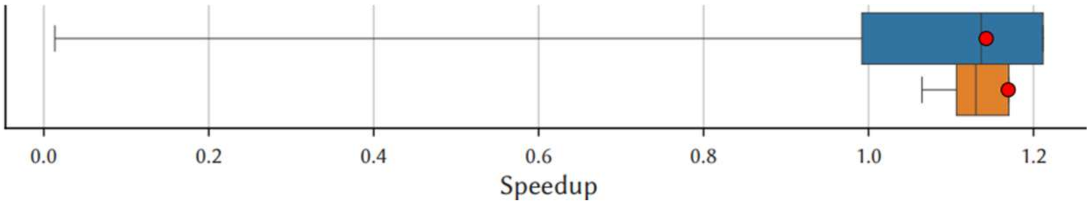
Up to 1.7x slower!

Hardware Upgrades

- For each software version,
  - Execute JOB on *Medium* machine
  - Execute JOB on *Large* machine
  - Compare speedup in going from *Medium* to *Large*

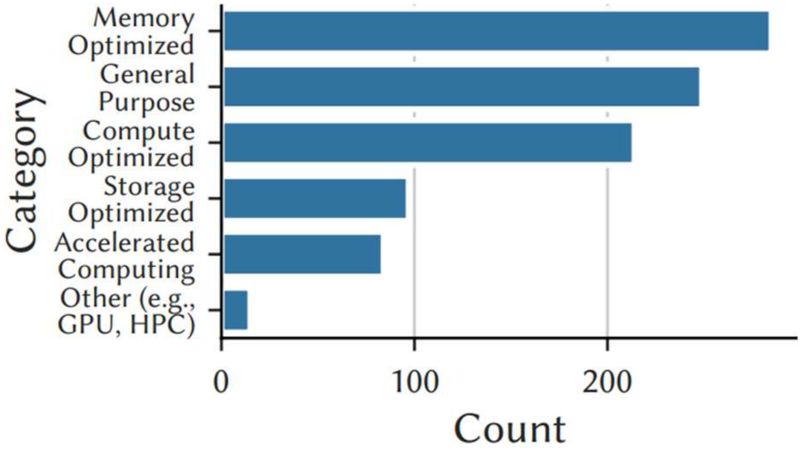


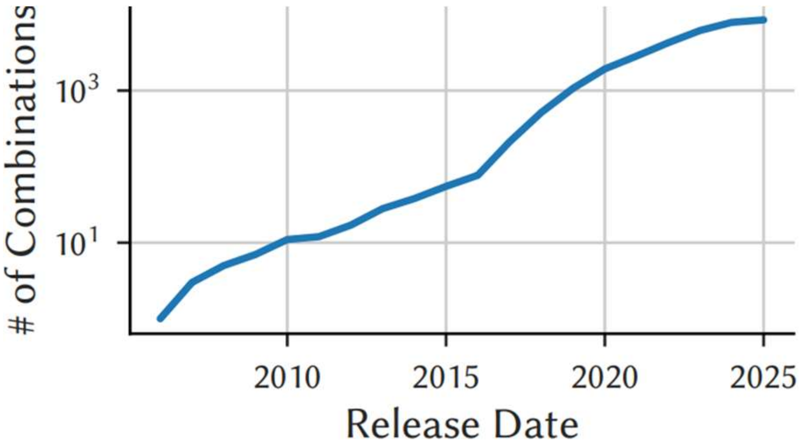
Query plan changes are risky!



Upgrading to the best setup

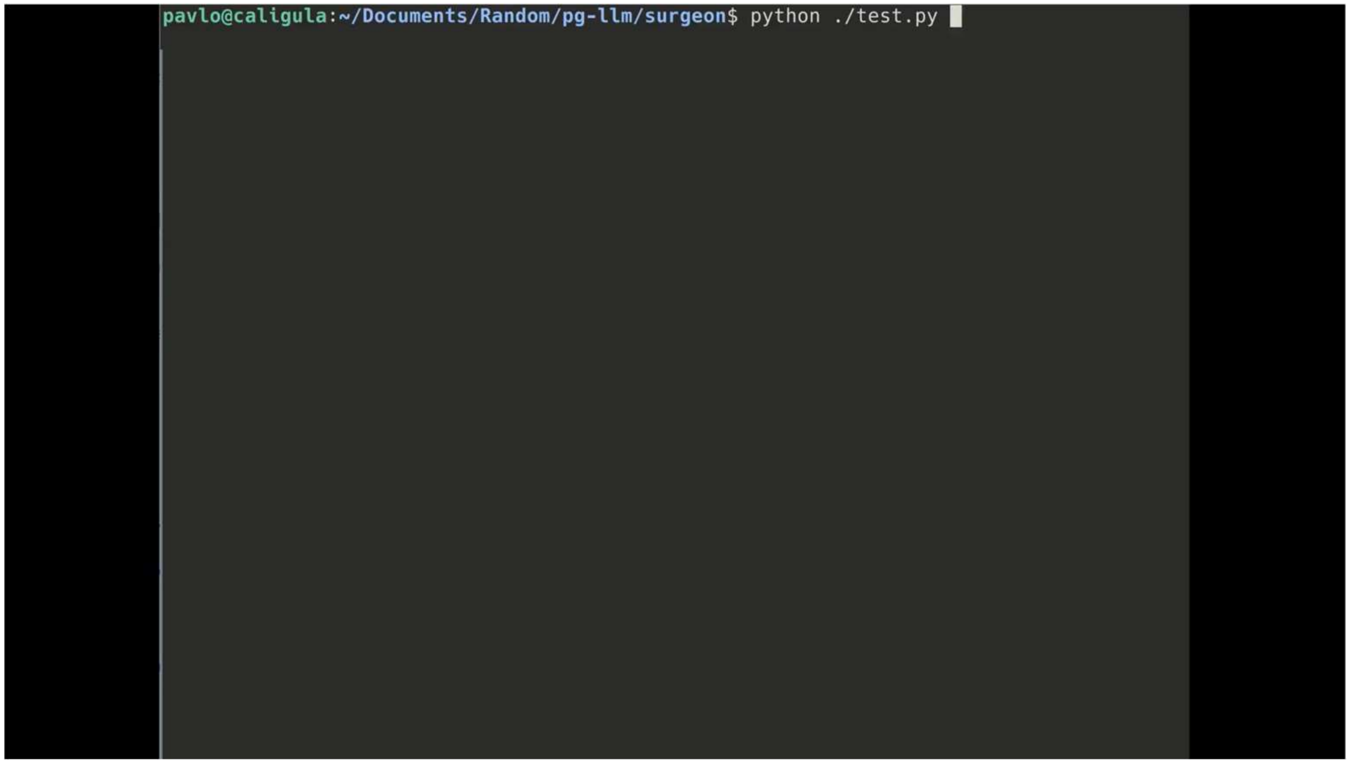
- {Software} X {Hardware} = too many possibilities to try them all!





Just Upgrade strategy does not scale!

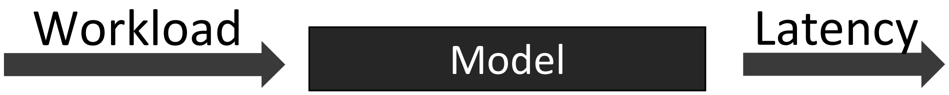
Ask an LLM?



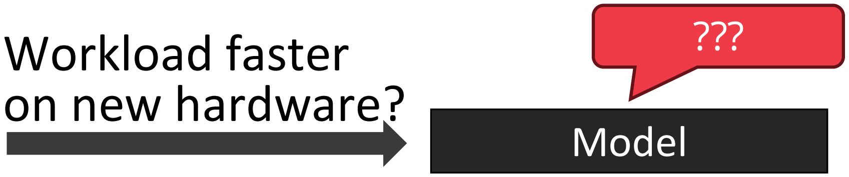
```
pavlo@caligula:~/Documents/Random/pg-llm/surgeon$ python ./test.py
```

Predicting DBMS Behavior

- Behavior models, $f: X \to Y$

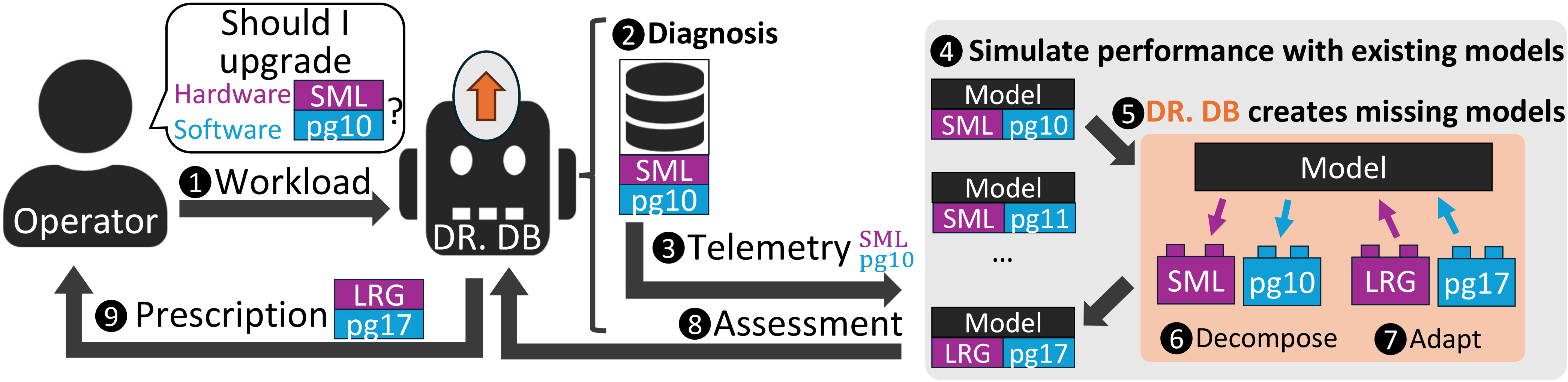


- Limitation: existing models assume **static** environments
  - Neither software- nor hardware-aware; can’t even ask the upgrade question!



Doctor Database

Prescribing database upgrades since 2025



Previously: pre-train an impractical number of models.
Now: DR. DB creates the necessary models at run-time!

Evaluation Setup

- Model: QueryFormer (Transformer-based)
- DBMS: PostgreSQL

3 workloads

- TPC-H (SF 10)
- DSB (SF 10)
- JOB

✖

9 software

- pg9
- …
- pg17

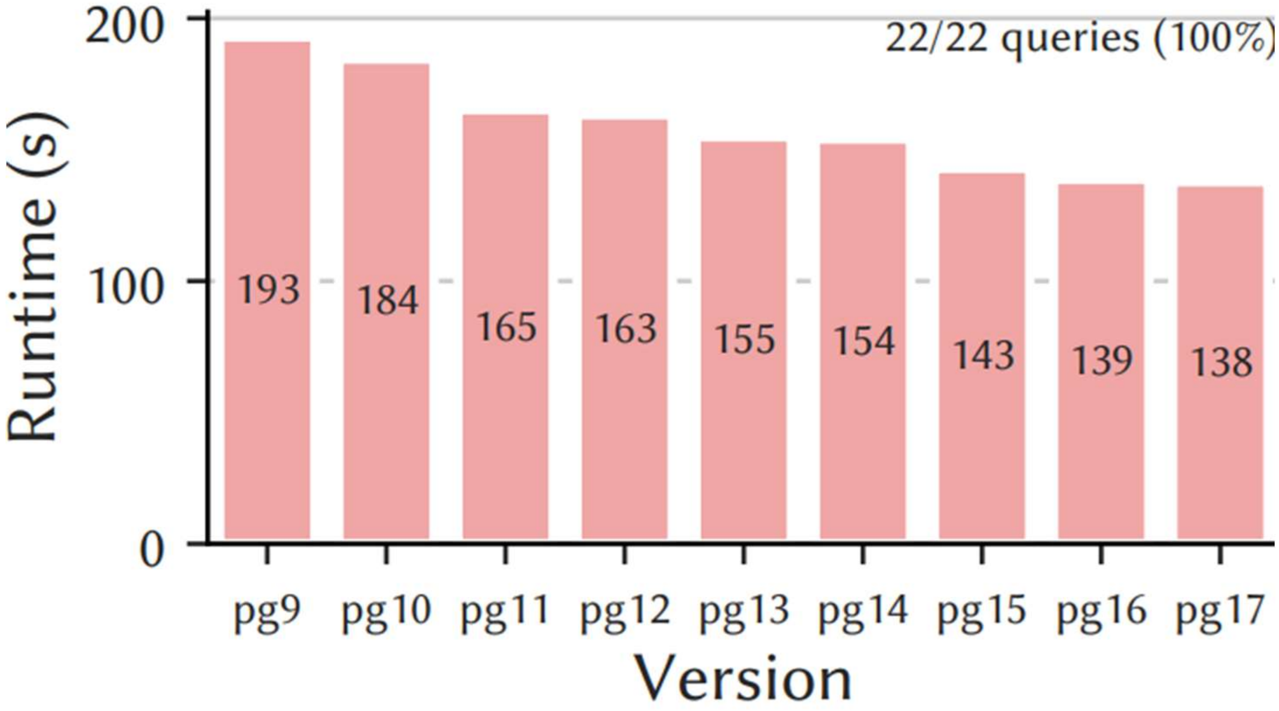
✖

3 hardware

- Small (SML)
- Medium (MED)
- Large (LRG)

* Omitted from talk

TPC-H Runtime: ~140s to ~190s

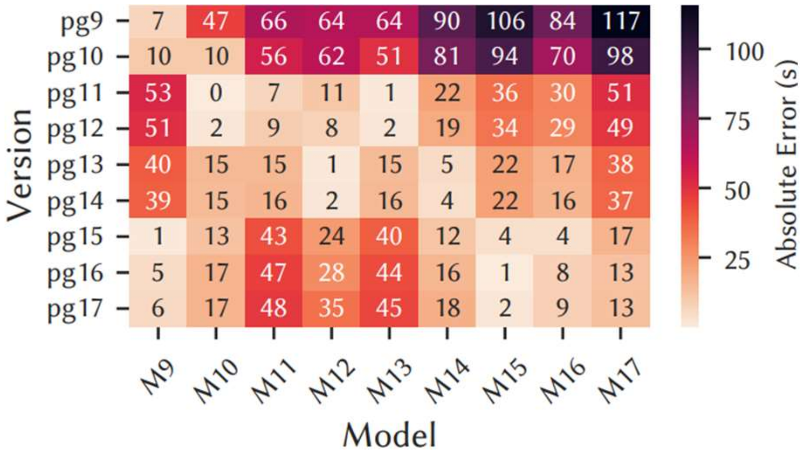


Doctor, how much faster?

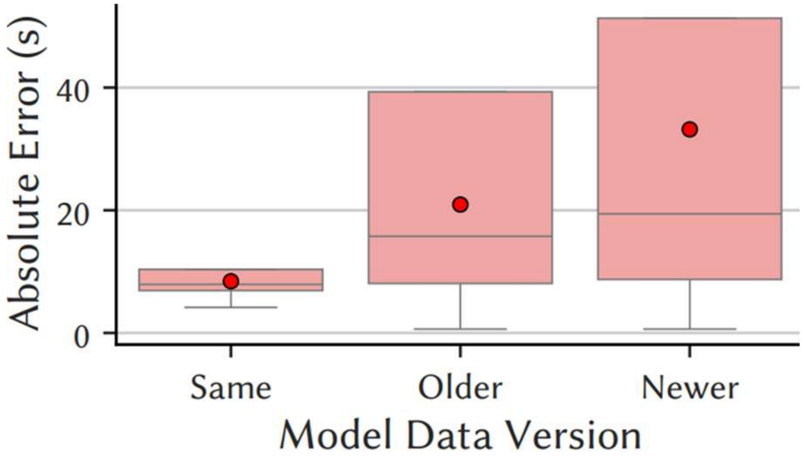
{Software, Hardware, Both}

- Metric: workload-level absolute error
- For each version pgN's training data, we train a separate model
    - Then for all training data versions, use each model to predict absolute run-time

Without model adaptation, up to 77x more error

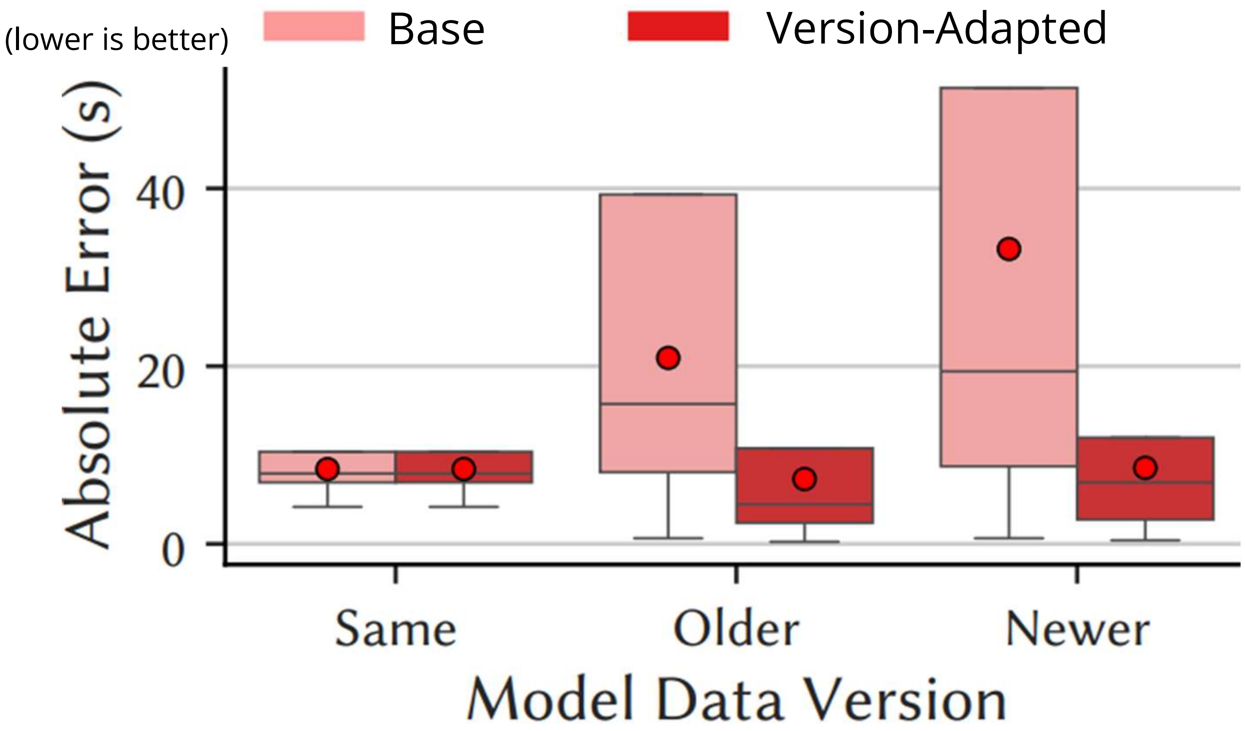


Version	M9	M10	M11	M12	M13	M14	M15	M16	M17
pg9	7	47	66	64	64	90	106	84	117
pg10	10	10	56	62	51	81	94	70	98
pg11	53	0	7	11	1	22	36	30	51
pg12	51	2	9	8	2	19	34	29	49
pg13	40	15	15	1	15	5	22	17	38
pg14	39	15	16	2	16	4	22	16	37
pg15	1	13	43	24	40	12	4	4	17
pg16	5	17	47	28	44	16	1	8	13
pg17	6	17	48	35	45	18	2	9	13



Version Adaptation

(workload run-time)



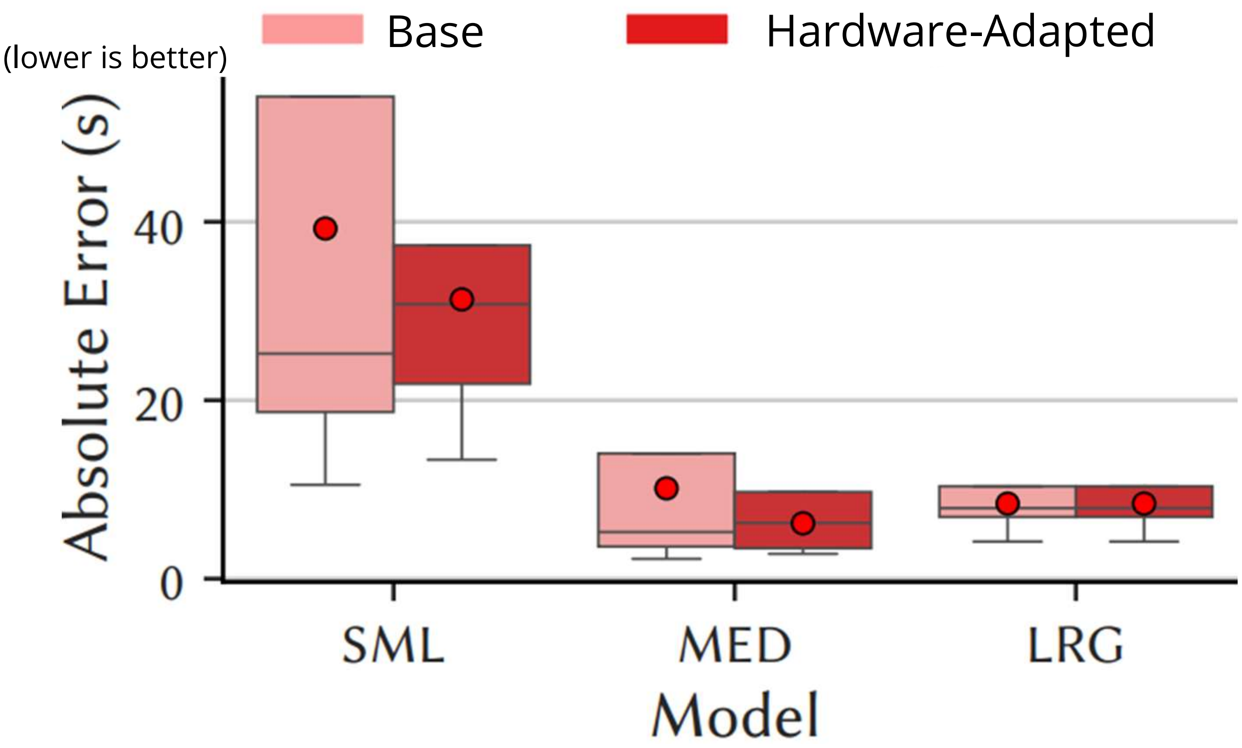
MEDIAN ERROR

Improvements up to:

- 4.3x (newer)
- 3.7x (older)

Hardware Adaptation

(workload run-time)



MEDIAN ERROR

Improvements up to:

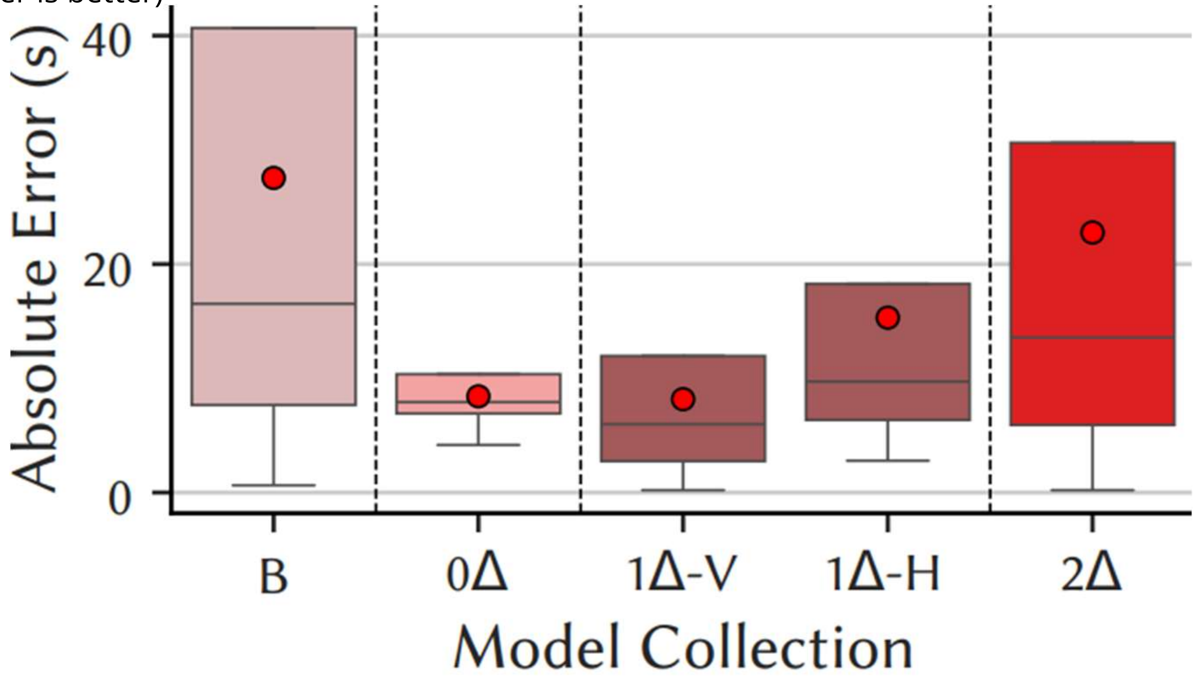
- 31x (LRG)
- 20x (MED)
- 6.6x (SML)

Multi-dimensional Δ-Adaptation

(workload run-time)

- Δ = how far is a model's training data from its evaluation scenario
- Akin to drawing from a probability distribution, Err ~ B vs Err ~ 0Δ

(lower is better)



Takeaways

- Doctor Database predicts performance impacts **pre-upgrade**
- It uses LoRA to make behavior models software- and hardware-aware
  - In a way that is composable and run-time swappable

- Part of broader Database Gym project (ML + DB)
  - Do we have to collect training data on all software and hardware combos? **No.**
  - Can we reuse expensive pre-trained models? **Yes, with some caveats.**

- Under submission, paper contains full details

✉ wanshenl@cs.cmu.edu

Ruiqi Wang ← on job market!
✉ ruiqiwan@cs.cmu.edu