



Will Angel

# Data Wrangling With DuckDB



# Key Points

**Fact:** DuckDB is an MIT-licensed embedded analytical database.

**Fact:** It's easy to go back and forth between DuckDB, Pandas, Polars through the magic of PyArrow

**Suggestion:** You should consider using DuckDB for your data processing and analysis.

**Argument:** You should *SERIOUSLY* consider trying DuckDB before going to “Big Data” tools





# About Me



1. I like ducks  
(wildlife photography)
2. I live in DC (Arlington, VA)
3. I help run meetups through Data  
Communiity DC (DC2.org)
4. I am a Lead Data Engineer @ Excella





# Agenda:

- What is DuckDB
- Why should you care about DuckDB
- When should you use DuckDB
- How can you use DuckDB
- Using DuckDB in Python
- Data processing performance and profiling
- A brief tangent on the shrinking of big data.





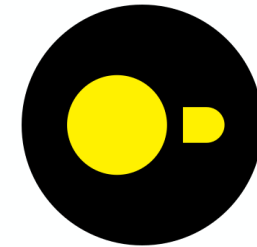
DuckDB Overview

# DuckDB: What, Why, When, How





# What is DuckDB?



# DuckDB

DuckDB is an open source fast in-process analytical database!


- Open Source: Free & Open – MIT License
- Fast: Performant. Quickly and efficiently runs analytical SQL queries
- In-process: Runs locally without a server
- Analytical: DuckDB is optimized for aggregations and analytical queries to support online analytical processing (OLAP). DuckDB supports ACID transactions, but is not as fast for online transaction processing (OLTP) workloads
- Database: DuckDB can be used to efficiently store relational data.






# Why should you care?

 DuckDB is a great tool for local SQL analysis. Import a file and you can do SQL locally!

 DuckDB is very ergonomic at data import / export, making it great for data pipelines and wrangling data

 DuckDB is starting to power the next generation of embedded analytical tools, so expect browser based data filtering tools to get more powerful.





# Why should you care (more technical)

- 🦆 DuckDB is like SQLite but for data processing. You can crunch significant amounts of data locally without spinning up a full database / data warehouse server, which may create significant time/cost savings and simplify system design
- 🦆 DuckDB is versatile and fast for data processing. Competitive with spark/polars in benchmarks.
- 🦆 DuckDB is portable with zero dependencies.







# Why you should care: PyData VA 25 update

*"Tools that build upon Apache Arrow are expected to yield significant reduction in non-value added engineering"*

- Will Ayd, Apache Arrow contributor & Pandas maintainer





# When should you use DuckDB

🦆 If you like SQL!

🦆 You have medium-largish data (~Gigabytes)

🦆 You don't want the hassle of procuring larger computing resources.





# How you can use DuckDb

- Directly in Python using the DuckDB package
  - Interchange with Pandas, Polars, Ibis, etc.
  - The magic of pyarrow!
  - As a backend for Data Apps (Streamlit, Gradio, etc)
- DuckDB CLI
  - Pure SQL
- DuckDB UI (experimental)
  - DuckDB –UI
  - Based on the motherduck UI.





Examples

# DuckDB versus Pandas



# A benchmark – group by aggregation:

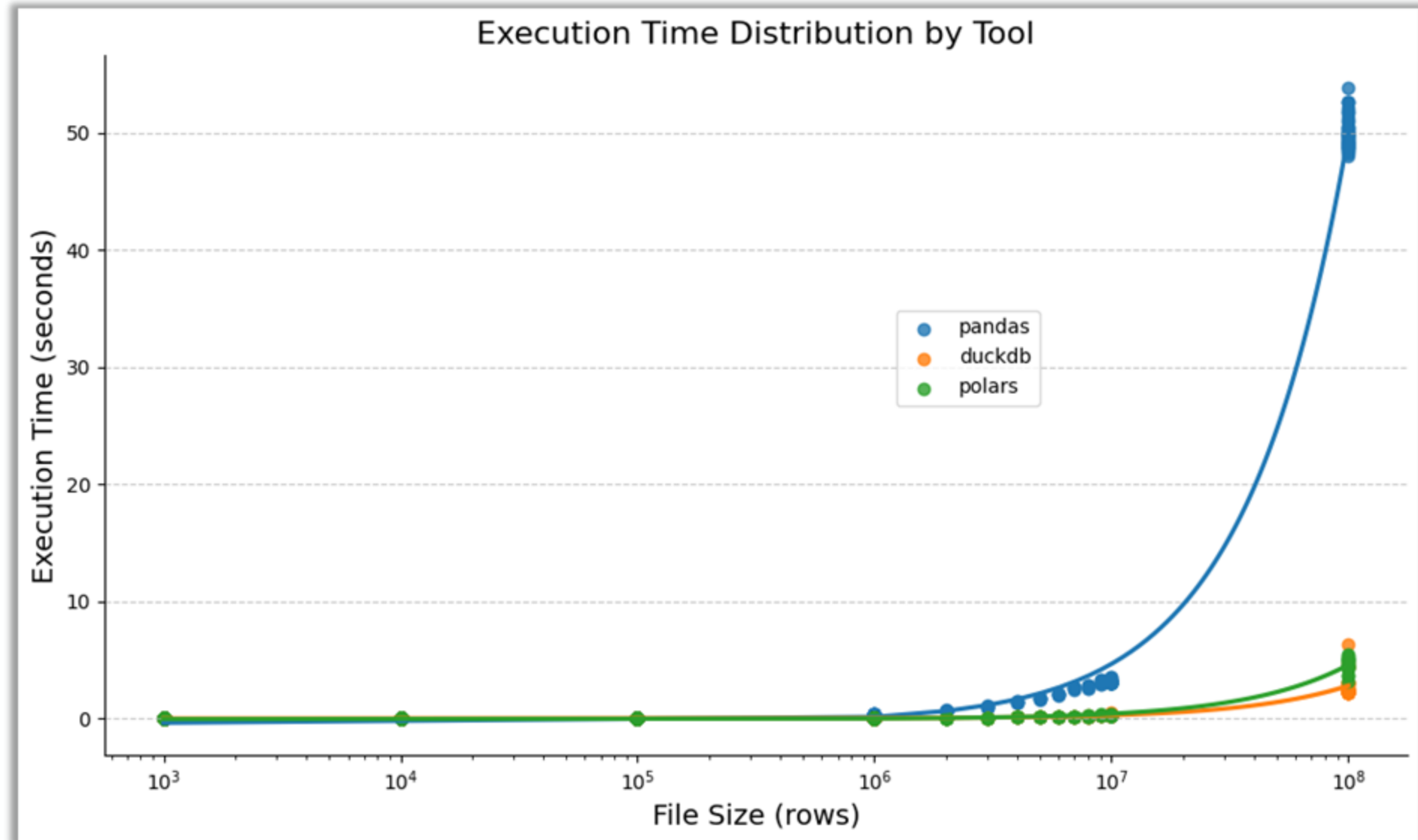


Benchmarking is hard

... But Pandas is almost always slower than modern tools like DuckDB and Polars

And that doesn't matter if Pandas is working well for you!

Use good / good enough tools!







Tangent

**Big Data:** smaller than it used to be





# DuckDB can efficiently utilize modern hardware

“I also benchmarked DuckDB on my 2021-era Macbook Pro. It has an M1 Pro CPU with 10 cores and 32 GB of RAM. I generated the TPC-DS data at 78 GB, 156 GB, and 313 GB scale, which approximately matches the scale-per-CPU of the other tested configurations.”

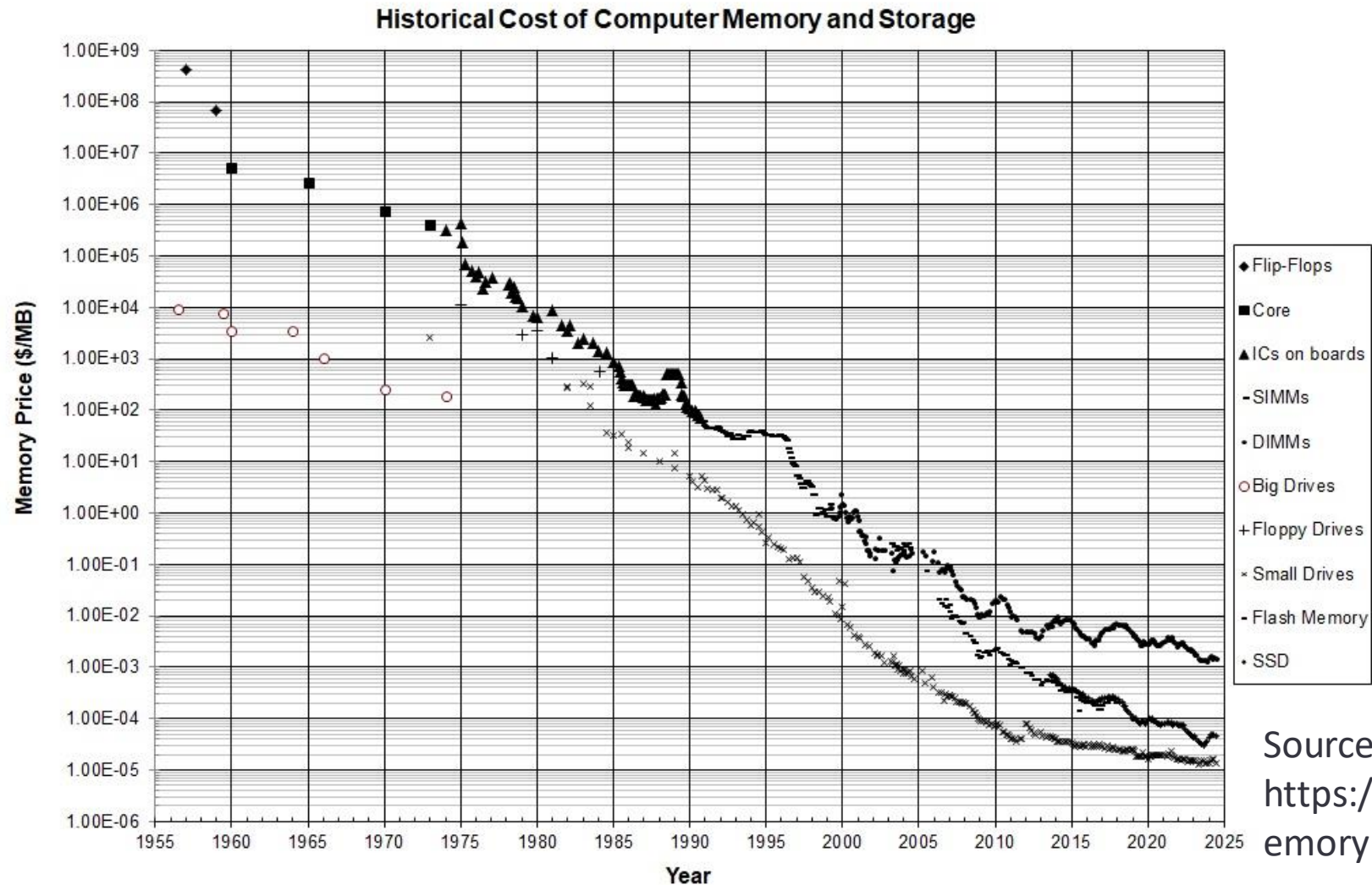
“The Macbook Pro is impressive. It outperforms a server with 16 cores. If you're an analyst with a recent Macbook Pro and a small data warehouse, **the laptop you use to write SQL queries might be faster than the data warehouse you run them on.**”

- George Fraser  
CEO, Fivetran



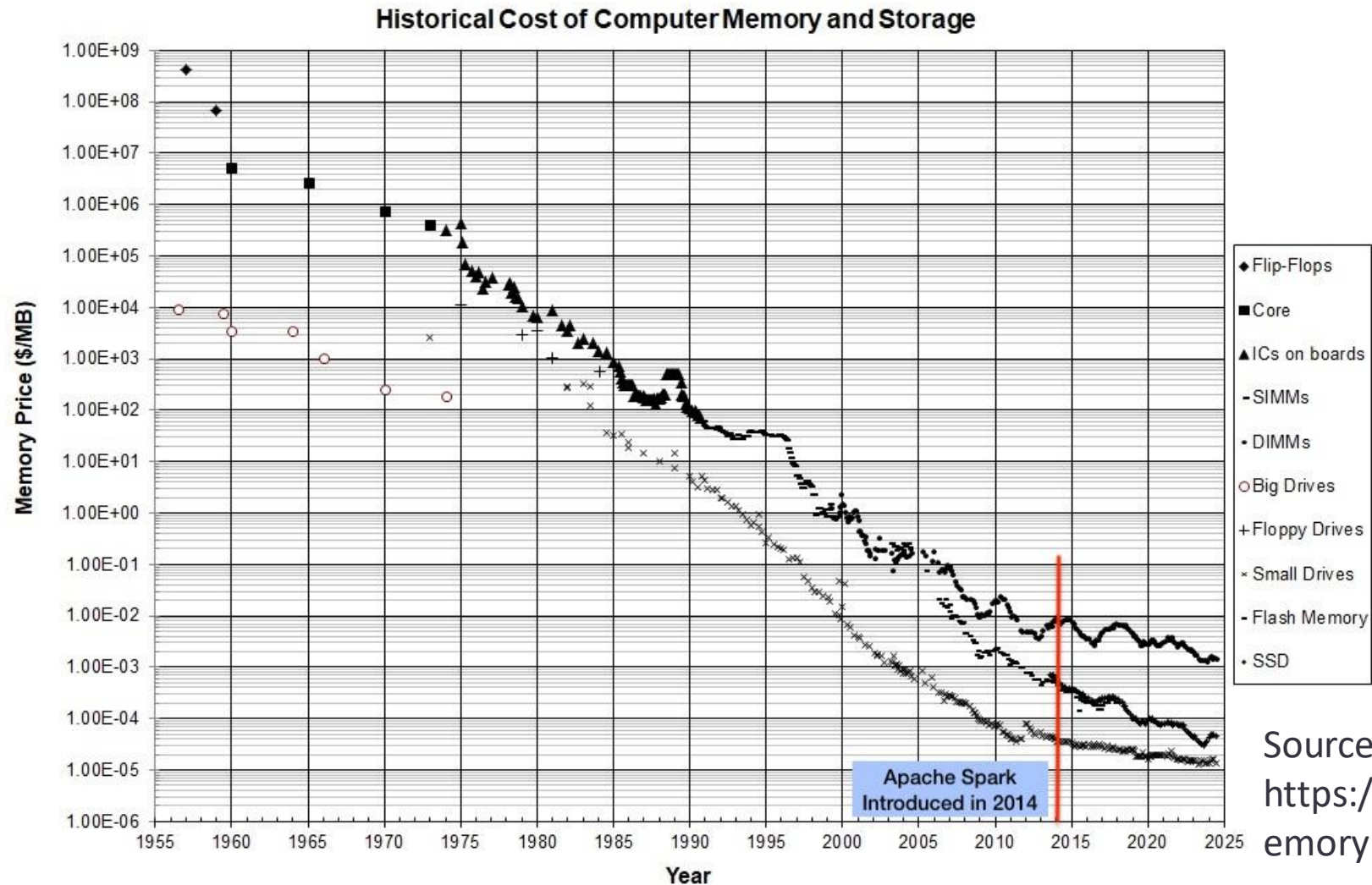


# Computers are getting bigger





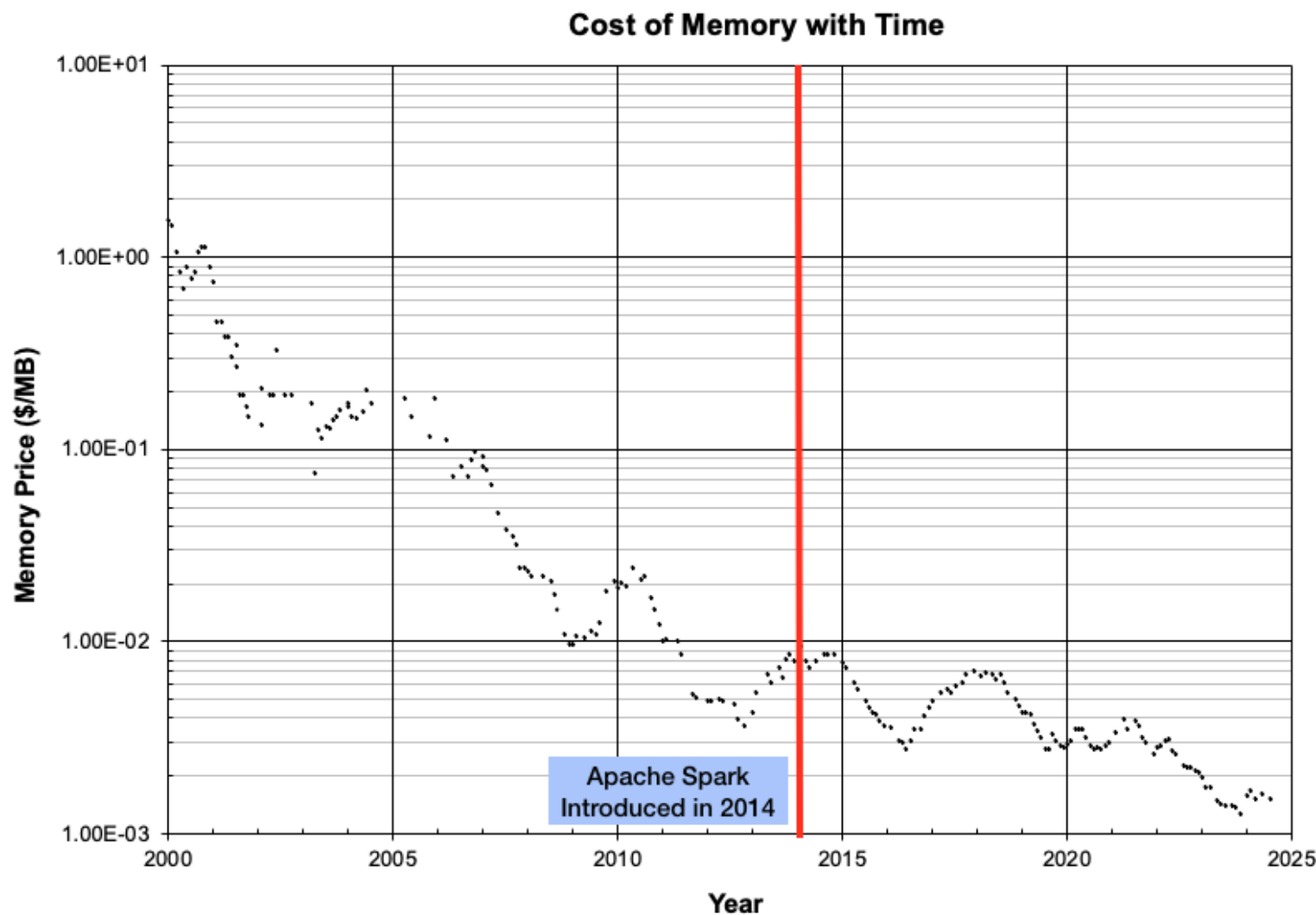
# Computers are getting bigger



Source:  
<https://jcmit.net/memoryprice.htm>



# Memory has gotten significantly cheaper

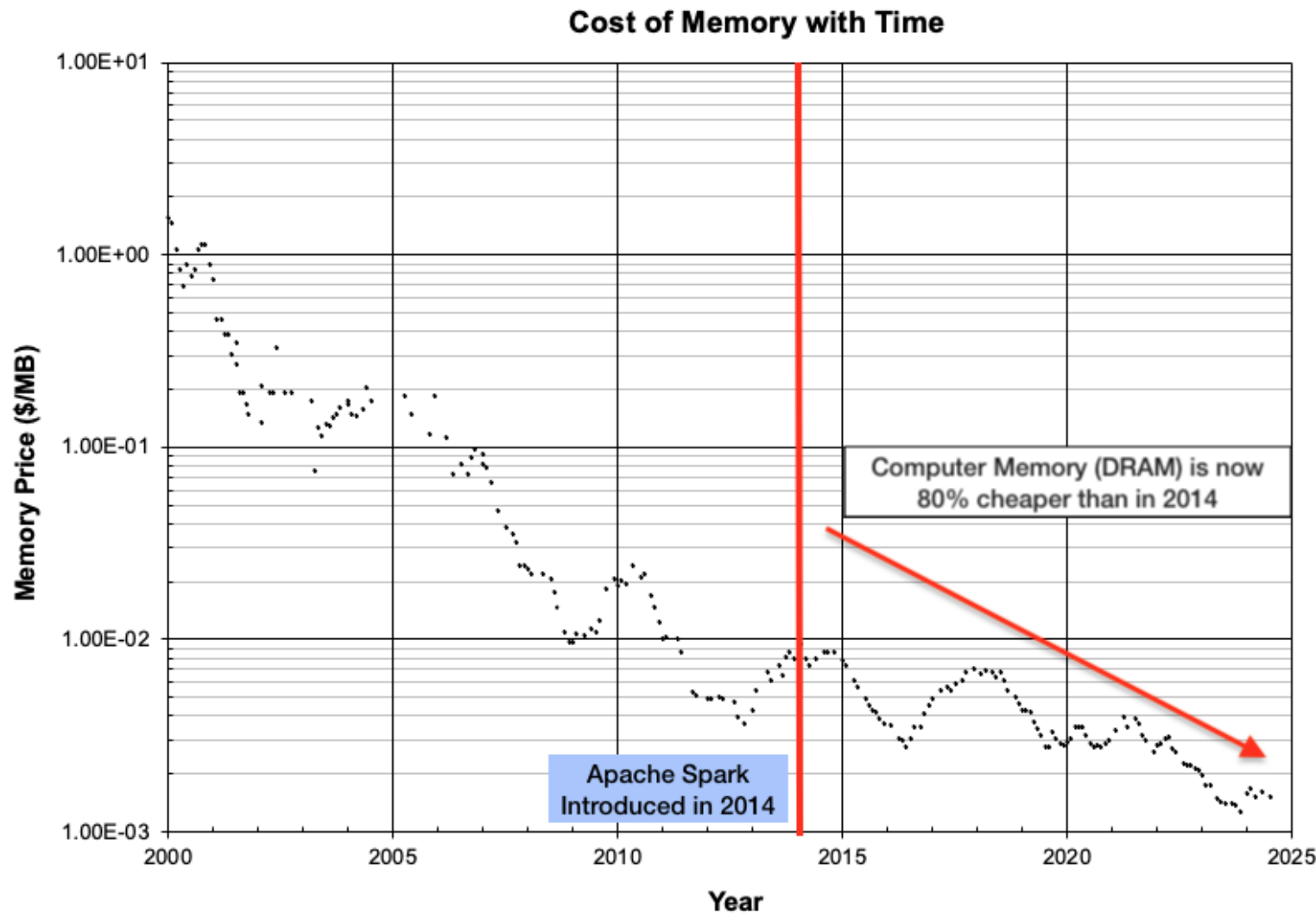


Source:  
<https://jcmit.net/memoryprice.htm>





# Memory has gotten significantly cheaper



Source:  
<https://jcmit.net/memoryprice.htm>





# Conclusion:

With more access to cheaper computing power and better tools (DuckDB/Polars/Ibis) that allow us to process data more efficiently:

1. We'll end up with more "Annoyingly Medium Data" - Too big to open in Excel, fits in RAM on a decent laptop
2. We can defer investing in large scale distributed systems like Spark more often

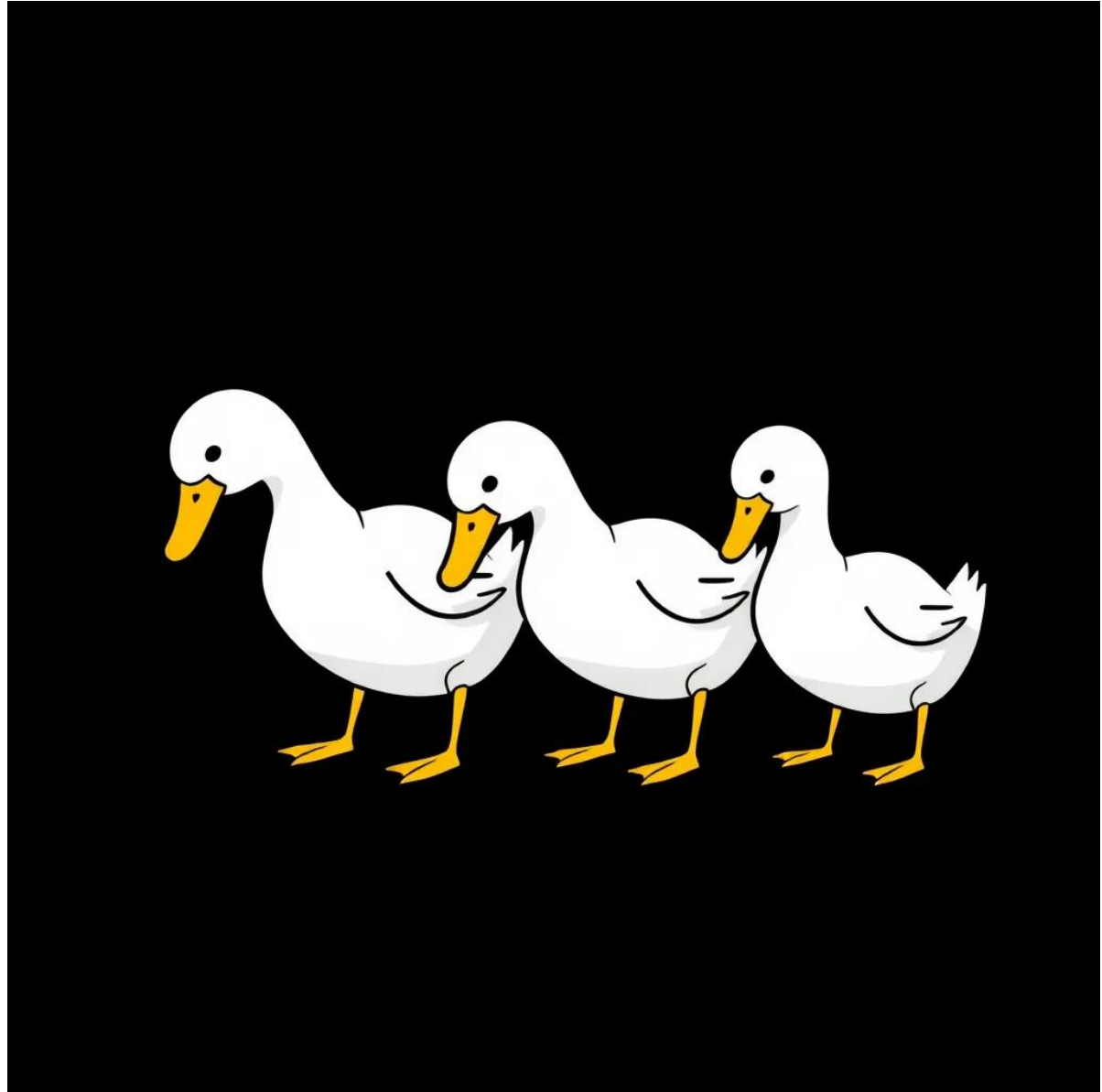


# Demos!

/in/william-angel/

@datadrivenangel

www.williamangel.net



AI ducks taking a bow: Flux-schnell Image Generation model. 