DuckDB

CWI
Database Architectures

1000 Days of DuckDB
Pareto Principle Still Holds

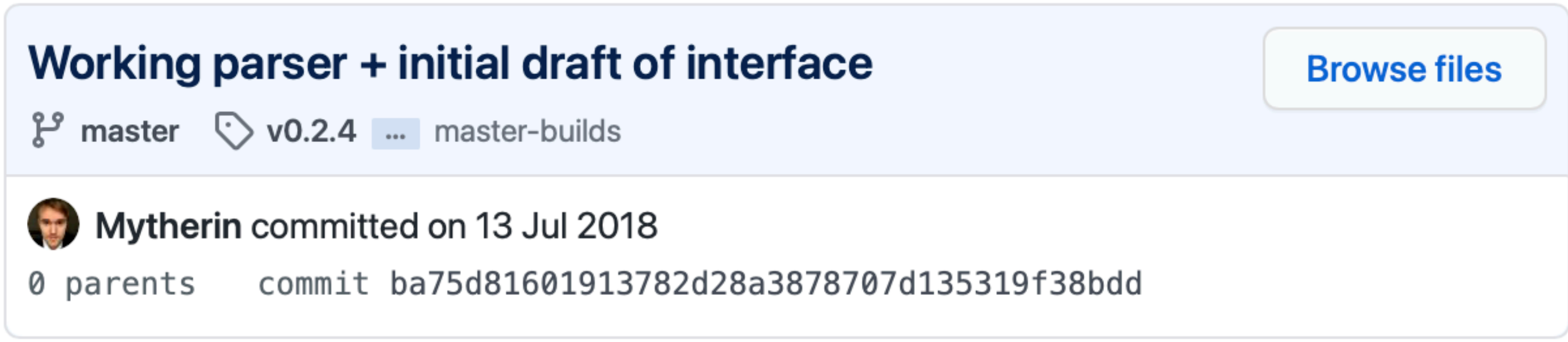
Hannes Mühleisen
Mark Raasveldt

DuckDB

- In-process OLAP DBMS
- Full SQL support
- No external dependencies
- APIs for C, C++, CLI, Python, R, Java, Node.JS, …
- Extensively tested
- MIT License

www.duckdb.org

D-Day 2018-07-13



Fridays are traditionally reserved for long shots at DA

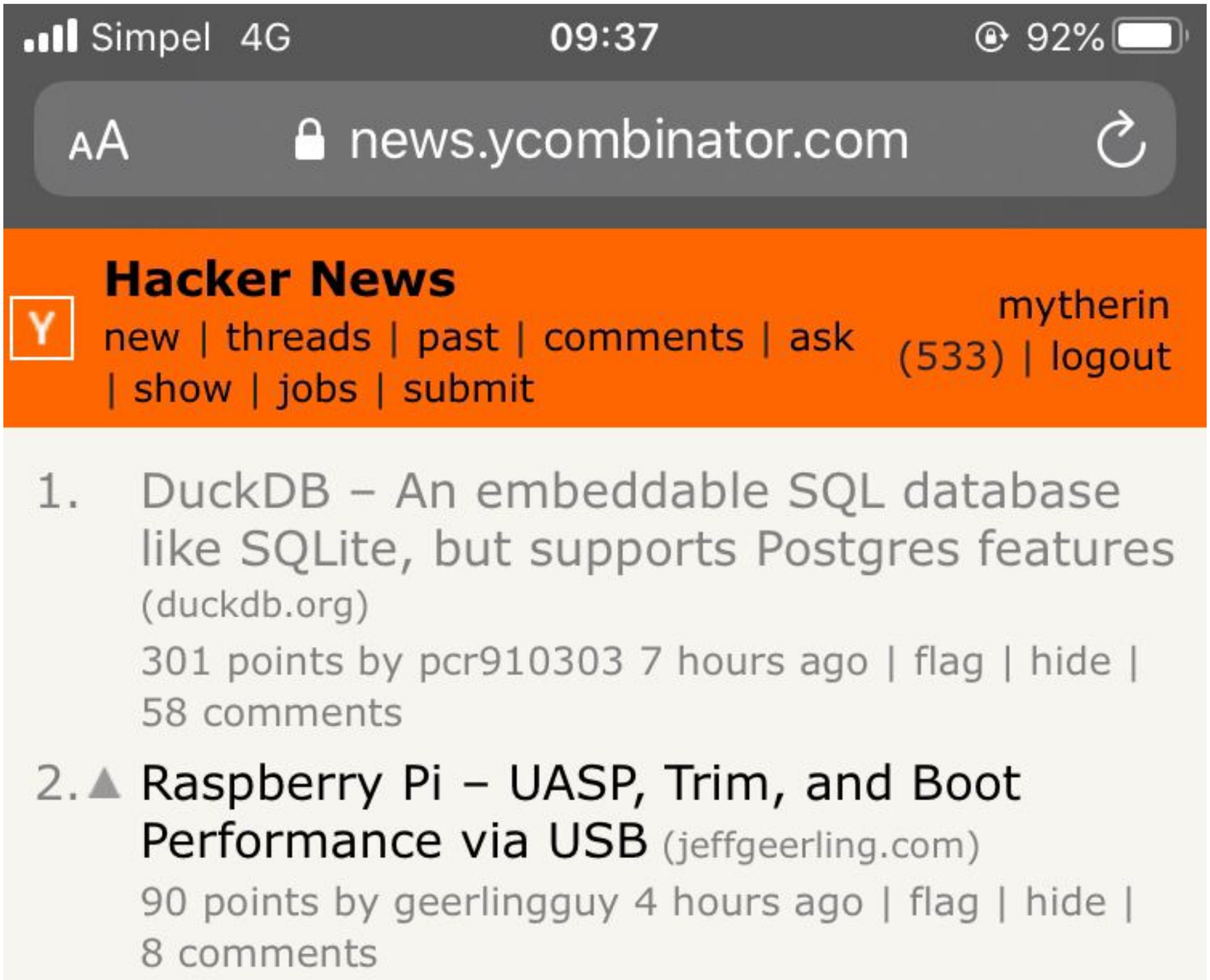
1000 Days of DuckDB

- ~ 6,500 Commits
- ~ 660 Pull Requests Merged
- ~ 600 Issues Closed
- ~ 2,600 Stars, ~ 166 Forks
- ~ 225,000 LOC (Amalgamation)

DuckDB Adoption

- Python: ~10,000 installations/month
- node.js: ~ 4,000 installations/month
- Java: ~2,500 installations/month
- R: ~2,000 installations per month
- + GH releases/clones, smaller platforms, …
- **~ 20,000 installations/month total**

Install Now!

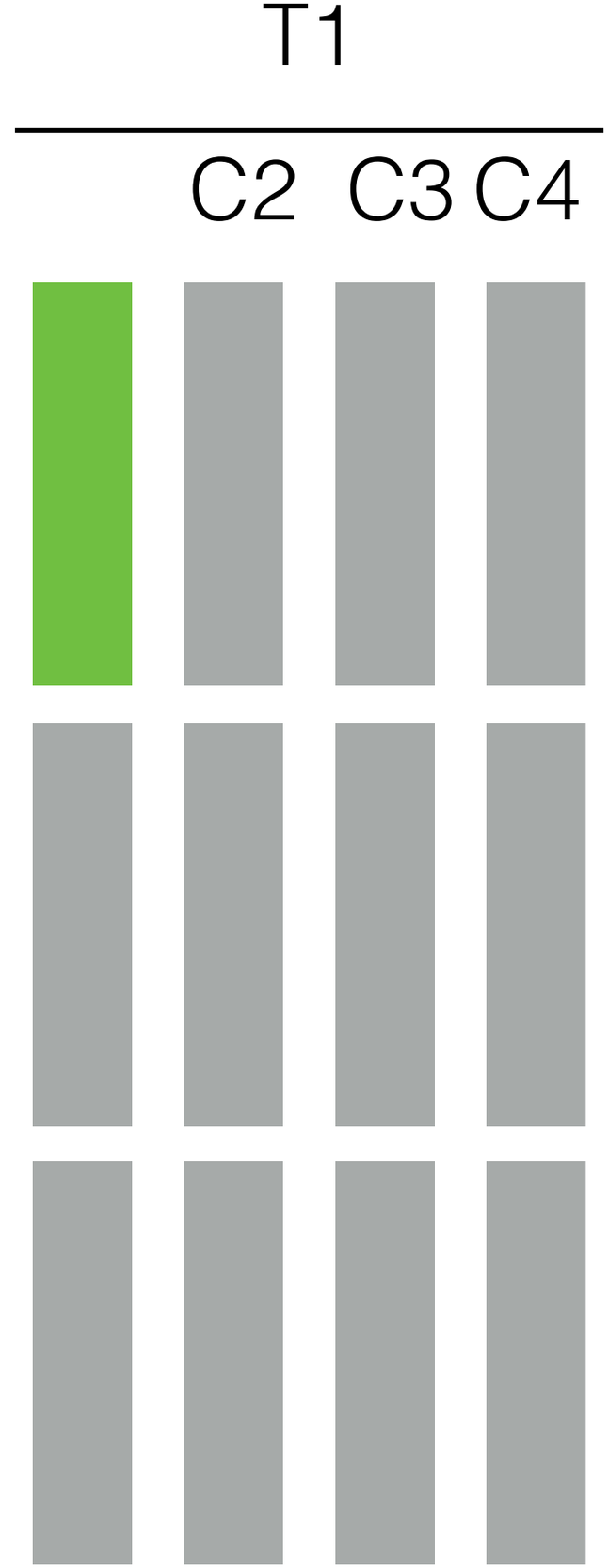

Simpel 4G
09:37
92%
AA
news.ycombinator.com
Hacker News
new | threads | past | comments | ask | show | jobs | submit
mytherin (533) | logout
1. DuckDB – An embeddable SQL database like SQLite, but supports Postgres features (duckdb.org)
301 points by pcr910303 7 hours ago | flag | hide | 58 comments
2. Raspberry Pi – UASP, Trim, and Boot Performance via USB (jeffgeerling.com)
90 points by geerlingguy 4 hours ago | flag | hide | 8 comments

Major Milestones

- **Inter-Pipeline Parallelism** for scans, aggregations etc.
- **Vector format** with compressed execution support and nested types
- OLAP-optimized **Multi-Version Concurrency Control**
- Lock-Free **Buffer Manager**
- **Statistics propagation**
- **Comprehensive SQL** including
  - Subquery folding
  - Recursive Common Table Expressions
  - Collations, Decimals, Window Functions, 100s of Scalar/Aggregate Functions…

Vector Format

- Vectors are core unit of data in DuckDB
  - “Vectorized engine”, “Vector volcano”
- Subset of a logical column of data
- Table: List of lists of vectors.
- Should support compression
- Should support nested types



Vector <> Vector

- Need compressed storage
  - Need to process compressed data
- Naive option: Decompress on Scan
  - Drawback: data & engine explosion

Too soon?

- **Better idea: Operate on compressed data directly**
- Challenge:
  - Many different compression methods
  - Many operators/functions
  - **Implementation effort / code size explosion**

Compression + Canonicalisation

- Vectors can stored in may encodings
  - Constants, plain, Dictionary, RLE, Bit packed, arbitrary expressions, …
  - Database operators are **free to implement each case**
    - Will be done for performance-critical ops
- But not every SQL function needs to, it can **canonicalise** the vector into common format
  - Whats the canonical format?

Canonical Vector Format

- Naive approach: Decompress into plain C-Style Array
  - Drawback: Slow, large intermediate
- Better: Use **“Orrification” canonicalisation** into
  - Define Mask (where are the NULLs)
  - Selection List (Offsets into data)
  - Data (plain values)
- **Free** for flat and constant vectors

Canonicalisation example: RLE

```
 10, 1
 42, 5
 43, 4
NULL, 1
 11, 1
```

Orrify()

Define	Offsets	Data
true	1	10
true	2	42
true	2	43
true	2	11
true	2	
true	2	
true	3	
true	3	
true	3	
true	3	
false	0	
true	4	

Paper coming!

OLAP MVCC

- Modified HyPer*
- Assume changes commit immediately write to in-memory table data
- ~~Keep undo version~~ **~~per tuple~~**
  - Huge storage overhead for batch updates
  - Retrieval needs version check on every tuple
- Keep undo versions per column chunk
  - OLAP updates bulky, column subset

```
DELETE
FROM orders
WHERE order_time
< DATE '2010-01-01'
```

```
UPDATE people
SET age=NULL
WHERE age=-99;
```

OLAP MVCC Benchmarks

- Example: Many single-column updates on table with **single** column

- Row-based MVCC unfit for use case

```
UPDATE tbl
SET i=i+1
```

SINGLE COLUMN

system	time (s)
hyper	24,0
monetdb	13,2
sqlite	8,9
duckdb	0,72

OLAP MVCC Benchmarks

- Example: Many single-column updates on table with **100** columns

- Row-based MVCC unfit for use case

```
UPDATE tbl
SET i=i+1
```

MANY COLUMNS

system	time (s)
hyper	149,3
monetdb	13,2
sqlite	155,3
duckdb	0,72

Paper coming!

Next-Gen System Goals

- Environment-Awareness
  - We are not the only one wanting to achieve something on this hardware
  - Allow checkpointing/resuming during queries
- Self-Driving (Hello, Andy)
  - No DBA
- Robustness
  - Environment breaks
  - e.g. CRC-during-scan, appends update checksum

2018

Next-Gen System Goals

- **Make it work first!**
- Environment-Awareness
  - We are not the only one wanting to achieve something on this hardware
  - Allow checkpointing/resuming during queries
- Self-Driving (Hello, Andy)
  - No DBA
- Robustness
  - Environment breaks
  - e.g. CRC-during-scan, appends update checksum

Next-Gen System Goals

- Environment-Awareness
  - We are not the only one wanting to achieve s hardware
  - Allow checkpointing/resuming during queries

Pointless in Isolation (Except for Paper)

- Self-Driving (Hello, Andy)
  - No DBA
- Robustness
  - Environment breaks
  - e.g. CRC-during-scan, appends update che

In-Process

SQLite

?

Stand-Alone

PostgreSQL

IBM DB2

TERADATA

ClickHouse

OLTP

OLAP

In-Process

Stand-Alone

SQLite

DuckDB

PostgreSQL

IBM DB2

TERADATA

ClickHouse

OLTP

OLAP

Next Steps

- Compressed Storage
- Cardinality Estimation
- Native UDFs for Python, R, etc.
- Finalize on-disk representation
  - Blocking release v1.0.0
- Make basic functionality **boring**
- Back to **interesting** issues from back in 2018

Very Special Thanks

- Peter Boncz (History Teacher)
- Till Döhmen (Automatic CSV Parsing)
- Orri Erling (Vectors)
- Denis Hirn (Recursive CTEs)
- Pedro Holanda (Indexing & more)
- Andre Kohn (WASM Compilation)
- Manuel Rigger (SQLancer)
- Richard Wesley (Aggregates)