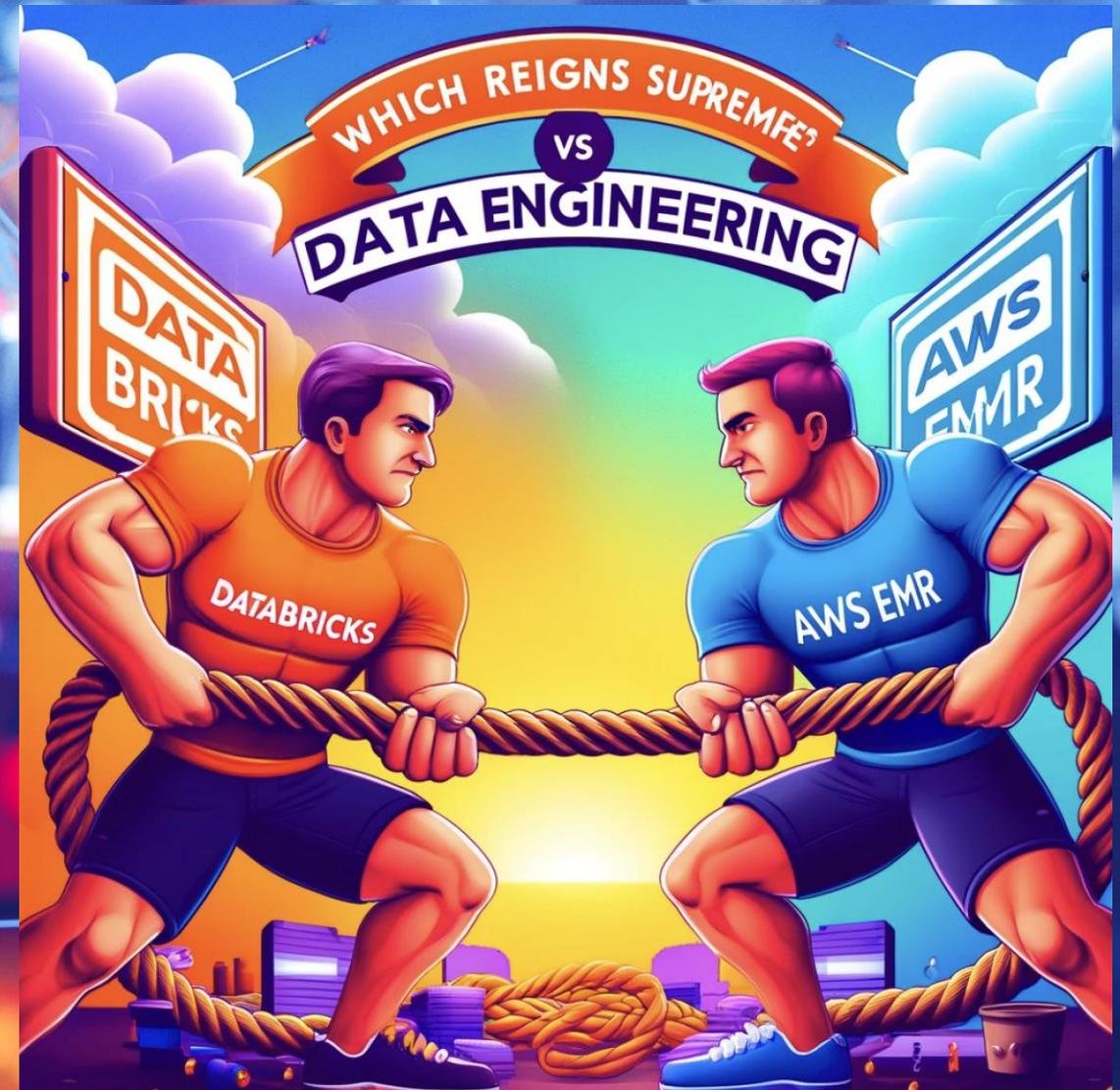
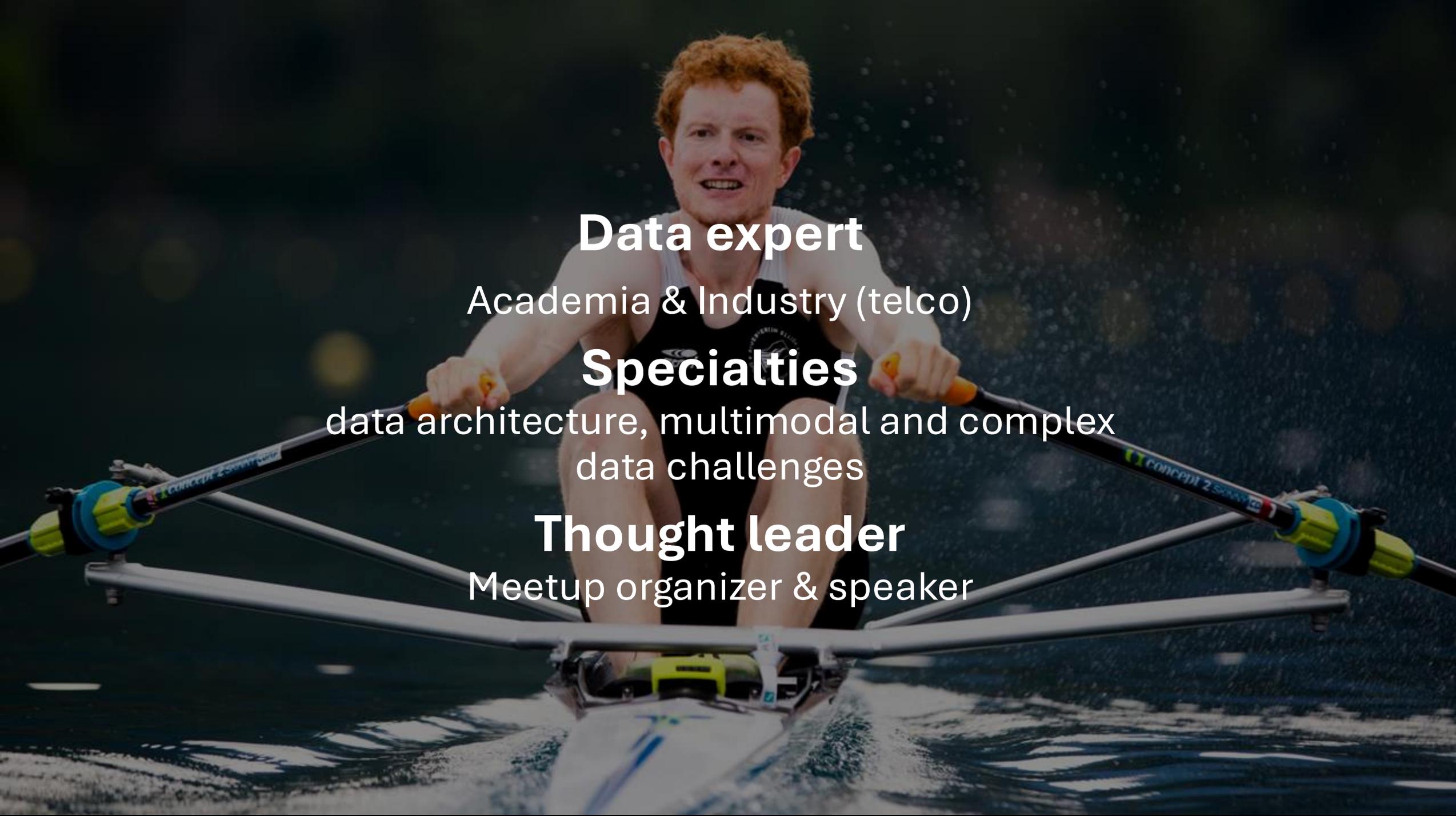


COST EFFICIENT ALTERNATIVE TO DATABRICKS

Georg Heiler



A man with red hair is rowing a boat on a body of water. He is wearing a black tank top and is captured in the middle of a stroke, with his arms extended forward holding the oars. The background is a blurred view of the water and sky. The text is overlaid on the center of the image.

Data expert

Academia & Industry (telco)

Specialties

data architecture, multimodal and complex
data challenges

Thought leader

Meetup organizer & speaker

ASci!

Supply Chain
Intelligence
Institute Austria

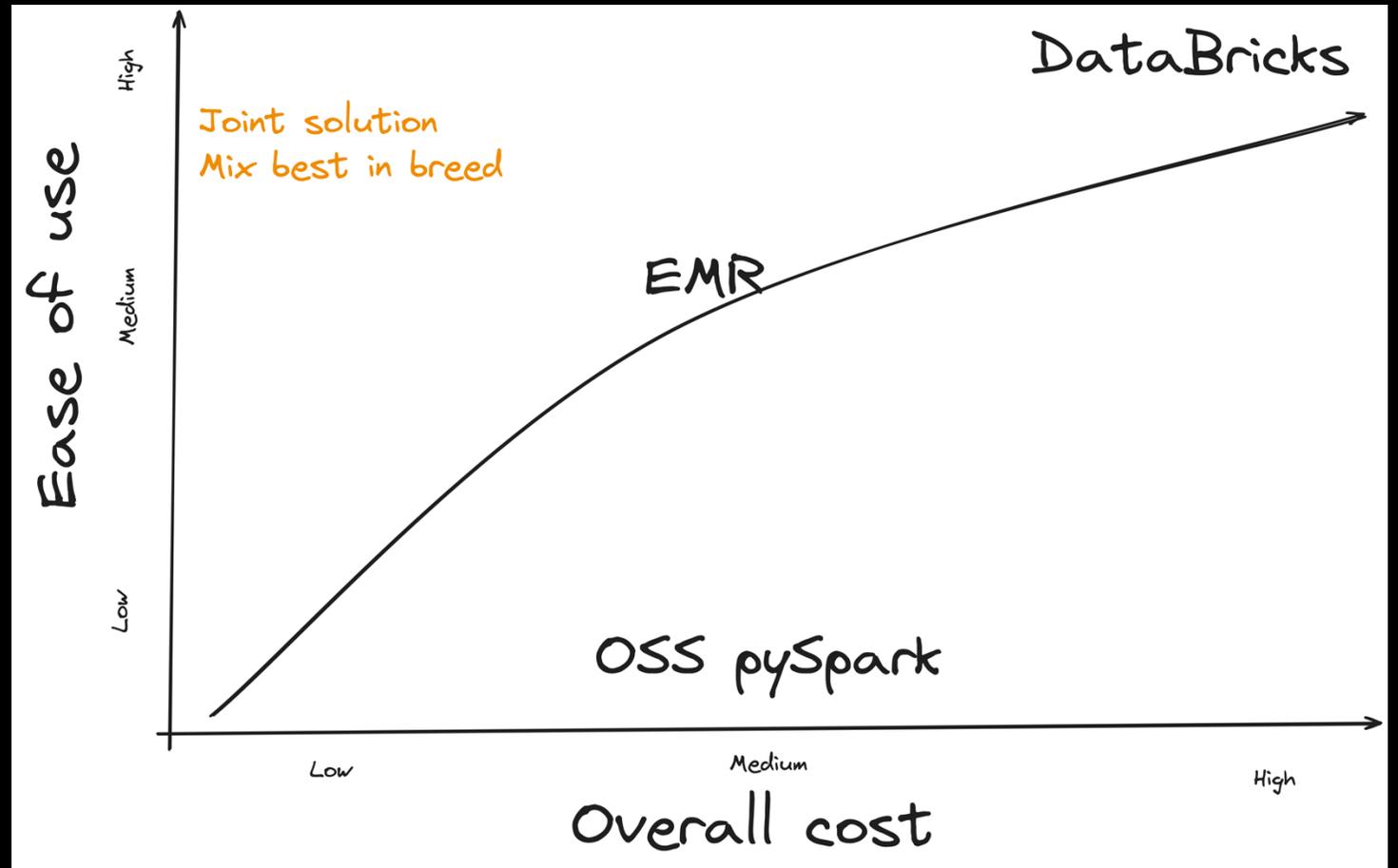
- Rising importance of understanding and shaping supply chains (covid, Ukraine war)
- No fine-grained clean data accessible
- Abundant un- and semistructured data → sophisticated cleaning & parsing required
- Extract and classify links based on semantic context

ASci!



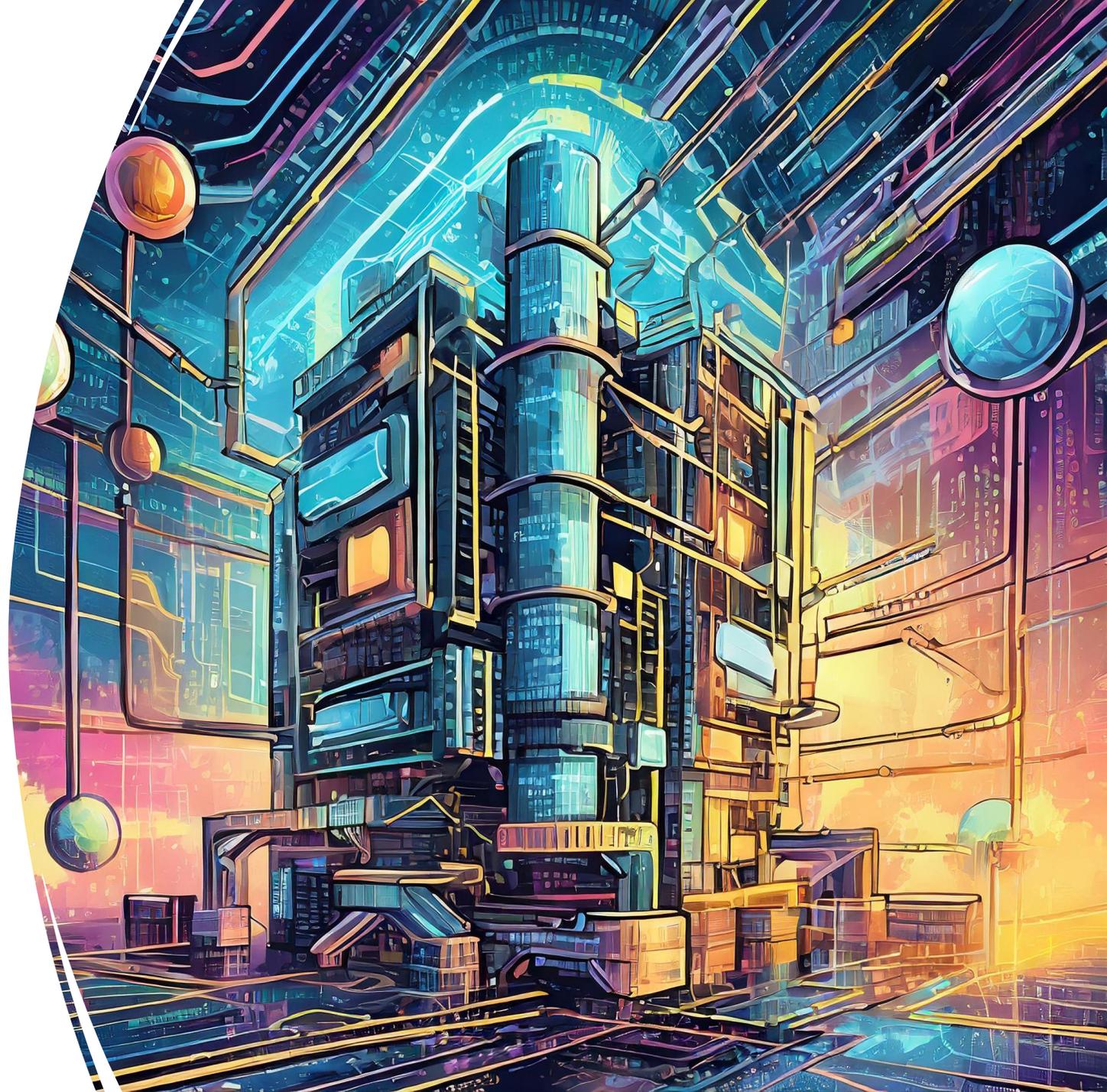
Results at a glance

- 43% Cost Reduction
- Software Engineering practices
- Future proof flexibility
- Single pane of glass for pipelines



History

- Mainframe
- Data warehouse
- Big Data (Hadoop)
- SQL on large data (Hive, Spark)
- Cloud DWH (Snowflake, bigquery)



PaaS offering

access control in platform metadata catalog

central platform

notebooks

orchestration

VCS integration

SQL access resource management

PaaS Solution Comparison

Databricks (DBR)

- Easy to use
- Can be expensive
- Lock-in features (permissions, catalog)
- Proprietary Photon engine

AWS Elastic Map Reduce (EMR)

- Price efficient
- Many tuning knobs available (& required)
- OSS Spark managed (scaled)

Challenges

- Runaway expenses (usage-based pricing)
- Missing software engineering best practices (notebooks)
- Developer productivity reduced
- Vendor lock-in

Vision

- 0-cost switch
- Software engineering practices
- Cost & lock-in reduction

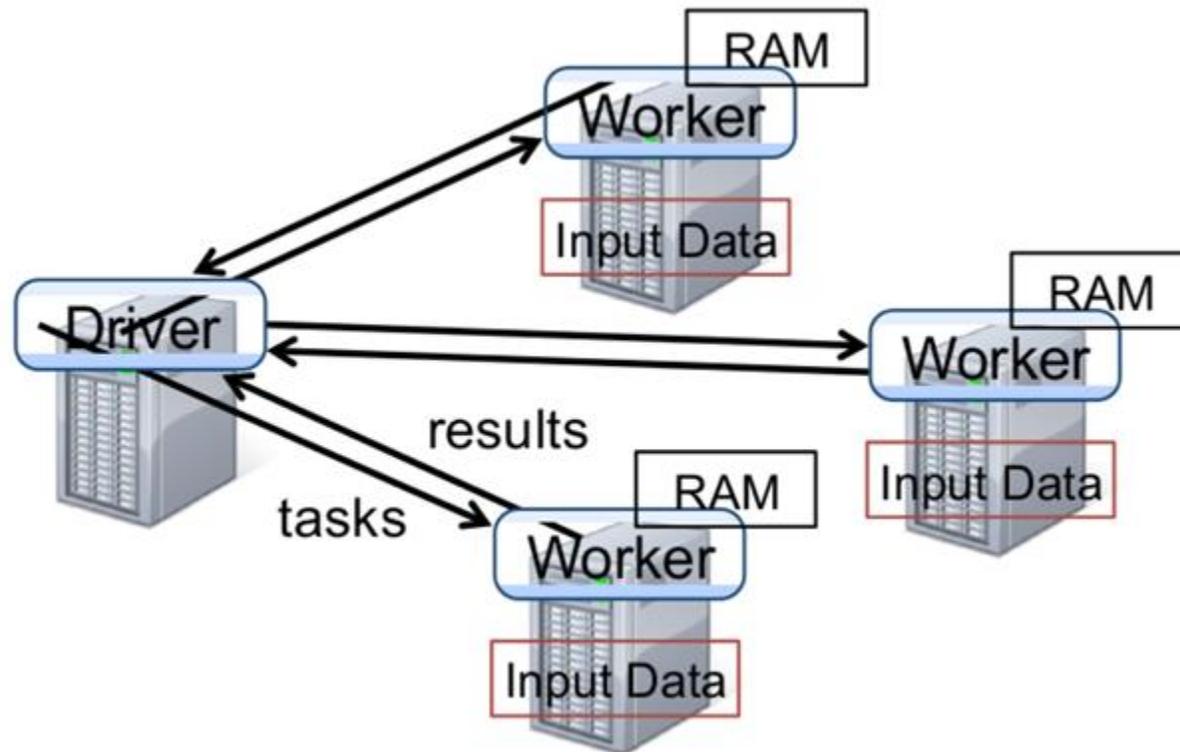
Orchestrator
(dagster)

Runtime
local

Runtime
remote DBR

Runtime
remote EMR

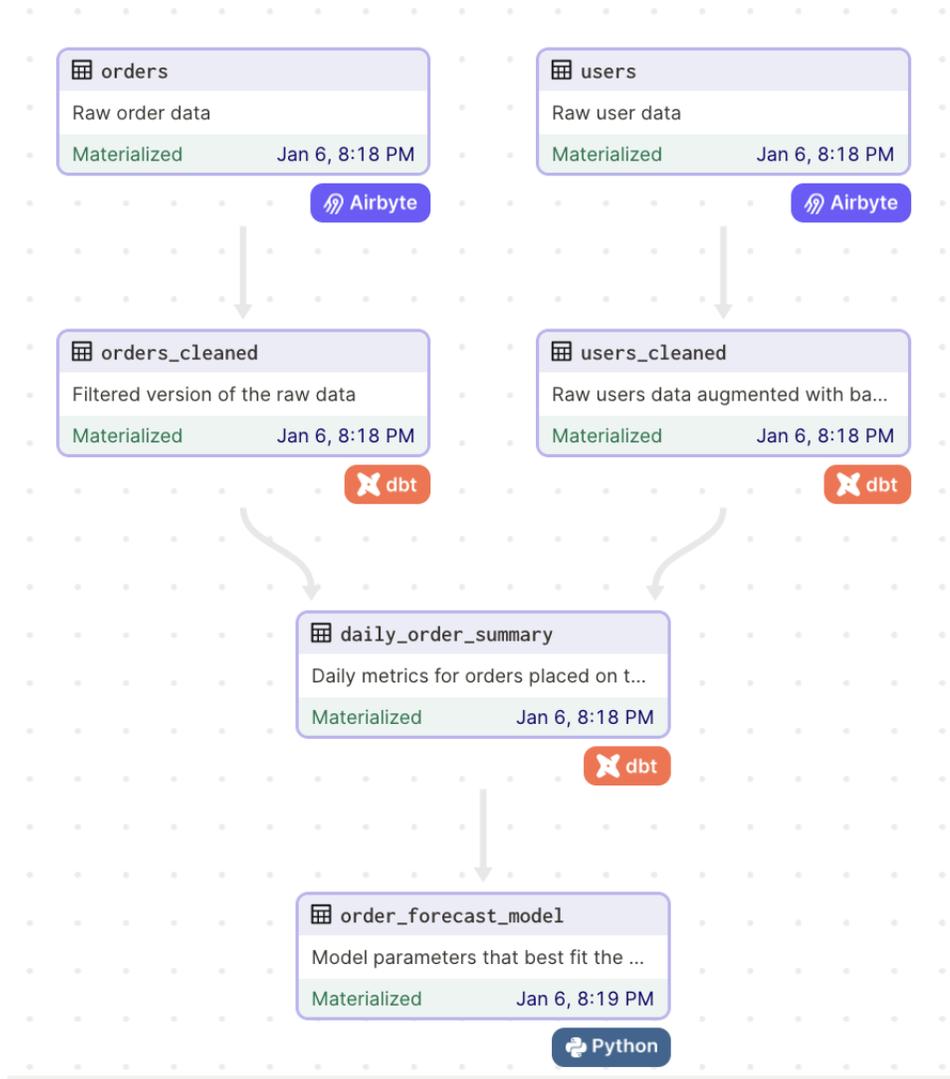
Spark at a glance



Dagster introduction



No distributed monolith of CRON strings
→ Asset aware event based orchestration



Observed challenges

- Remote execution
 - Parameter injection
 - Logging
 - Opaque SaaS tools
 - Single pane of glass
 - Dependency bootstrap
 - Missing testability in notebooks
- Large-scale compute & orchestrator native development

Orchestrator
(dagster)

Runtime
local

Runtime
remote DBR

Runtime
remote EMR

Dagster-pipes

What is Pipes

Launches with parameters and context info
(e.g. `partition_key`)

Orchestration Process

Imports Dagster
Can access instance

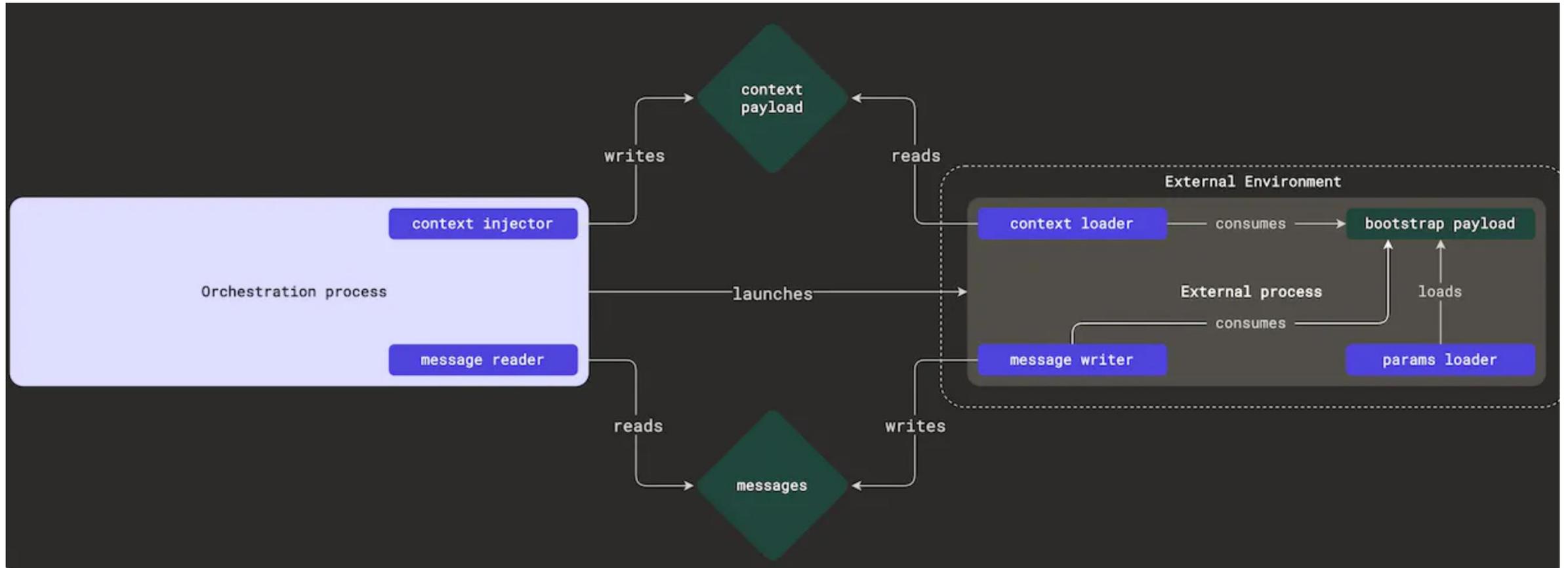


External Process

Lightweight dependencies
Minimal code changes

Streams logs and standardized metadata
to filesystem/s3/etc.

Dagster-pipes - Architecture



Dagster-pipes - Sample

External code (with metadata)

```
def main():
    orders_df = pd.DataFrame({"order_id": [1, 2]})
    total_orders = len(orders_df)
    context = PipesContext.get()
    print(context.get_extra("foo"))
    context.log.info("Here from remote")
    context.report_asset_materialization(
        metadata={"num_orders": len(orders_df)}
    )

if __name__ == "__main__":
    with open_dagster_pipes():
        main()
```

Internal asset shim orchestrating the execution of external script

```
@asset
def subprocess_asset(
    context: AssetExecutionContext, pipes_subprocess_client: PipesSubprocessClient
) -> MaterializeResult:
    cmd = [shutil.which("python"), file_relative_path(__file__, "external_code.py")]
    return pipes_subprocess_client.run(
        command=cmd,
        context=context,
        extras={"foo": "bar"},
        env={
            "MY_ENV_VAR_IN_SUBPROCESS": "my_value",
        },
    ).get_materialize_result()
```

Results & Demo



Overview Runs Assets Jobs Automation Deployment

5196d88 Success Run of example_pipeline @ b73a177b 2 assets 6. Sept., 09:41:32 0:01:18 View job View

example_s3__mini_example__s: example_s3__mini_example__s:

Re-execute ("example_s3__min

Preparing (0)
No steps are waiting to execute

Executing (0)
No steps are executing

Errored (0)
No steps have errored

Succeeded (2)

* "example_s3__mini_example__step1" Hide unselected steps

Events stdout stderr query:"example_s3__mini_example__st" Hide non-matches Levels (6)

TIMESTAMP	OP	EVENT TYPE	INFO
09:41:32,108	example_s3__mini_example__step1	STEP_WORKER_STA...	Launching subprocess for "example_s3__mini_example__step1".
09:41:32,683	example_s3__mini...	STEP_WORKER_STA...	Executing step "example_s3__mini_example__step1" in subprocess.

```

.. INFO [pipes] external process successfully opened dagster pipes.
.. INFO Partition key: None
ni_example__step1 Execution mode: ExecutionMode.SmallDevSampleLocal

```

Demo: youtube.com/watch?v=W27C5LpdEkE

Partitioned UI

> aws_s3 / commoncrawl / nodes_only

Events Plots Definition Lineage Automation

0 nodes Selected

nodes	IID	crawl
sample_seeds_1	●	CC-MAIN-2023-50
	●	CC-MAIN-2023-40
	●	CC-MAIN-2023-23
	●	CC-MAIN-2023-14
	●	CC-MAIN-2023-06
	●	CC-MAIN-2022-49
	●	CC-MAIN-2022-40
	●	CC-MAIN-2022-33
	●	CC-MAIN-2022-27
	●	CC-MAIN-2022-21
	●	CC-MAIN-2022-05
	●	CC-MAIN-2021-49
	●	CC-MAIN-2021-43
	●	CC-MAIN-2021-39
	●	CC-MAIN-2021-31
	●	CC-MAIN-2021-25
	●	CC-MAIN-2021-21
	●	CC-MAIN-2021-17
	●	CC-MAIN-2021-10
	●	CC-MAIN-2021-04
	●	CC-MAIN-2020-50

sample_seeds_1|CC-MAIN-2023-50

Materialized

Latest materialization

3. Feb., 07:58

Run

e1030285

Job

cc_crunching @ 5b77t
aws_s3_commoncra

Metadata

seed_node_count	100
cc_cnt	0
cc_warc_cnt	0
relative_overall_per_warc	0
timing_optimize_overall_minutes	0.16296786069869995
timing_optimize_no_deletion_minutes	0.04979061285654704

Source data

No upstream materializations to display.

System tags

code_version	e1030285-8e9a-411a-94e8-5e57cfe6fd93
data_version	1.0.0
data_version_is_user_provided	true
partition/a_seednodes	sample_seeds_1
partition/crawl	CC-MAIN-2023-50

Hide tags

Launch runs to materialize aws_s3 / commoncra

Partition selection

IID a_seednodes

Select partitions to materialize.

sampl...eds_1

+ Add a partition

IID crawl

Select partitions to materialize.

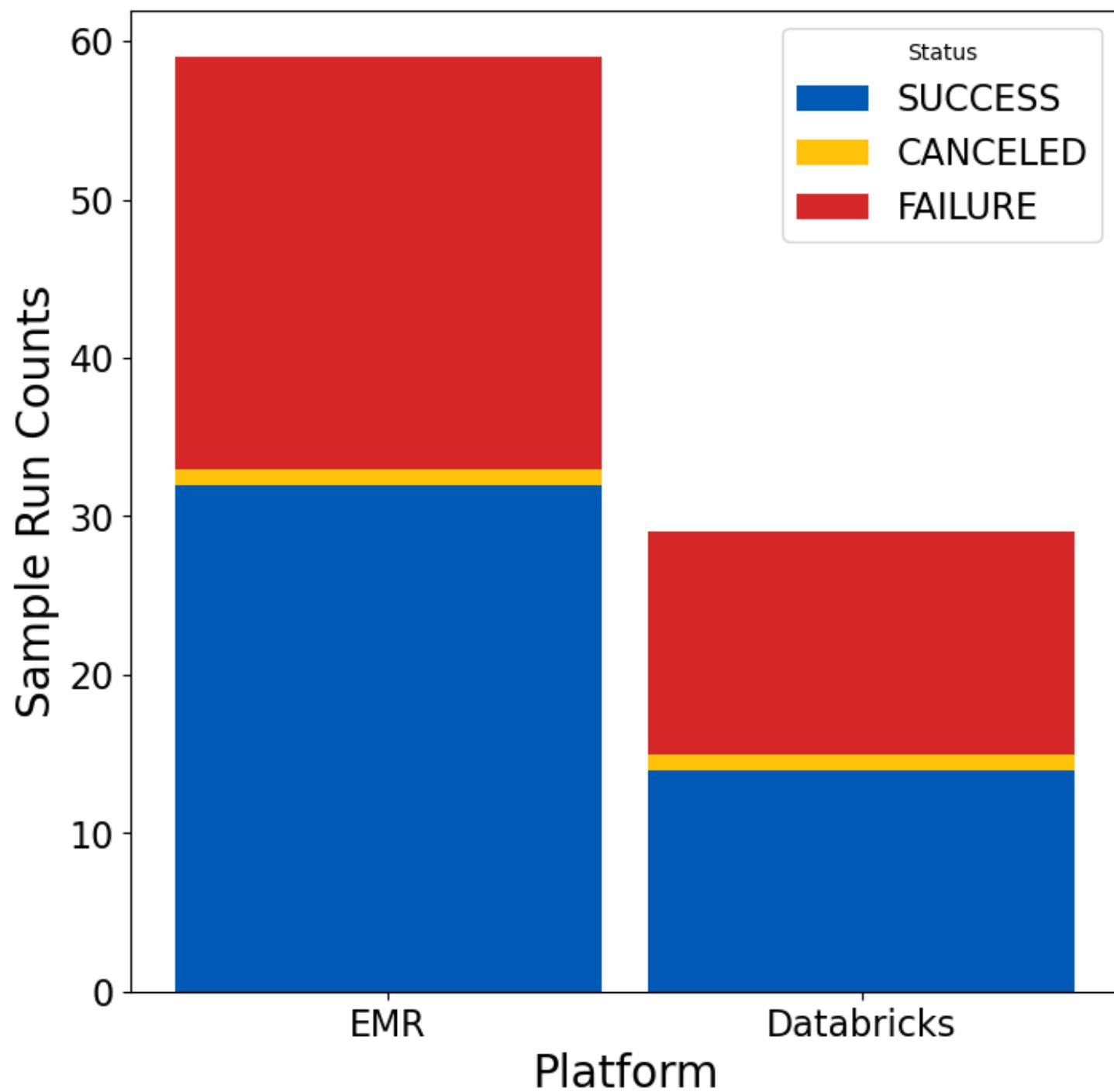
CC-MAI...023-50

Tags

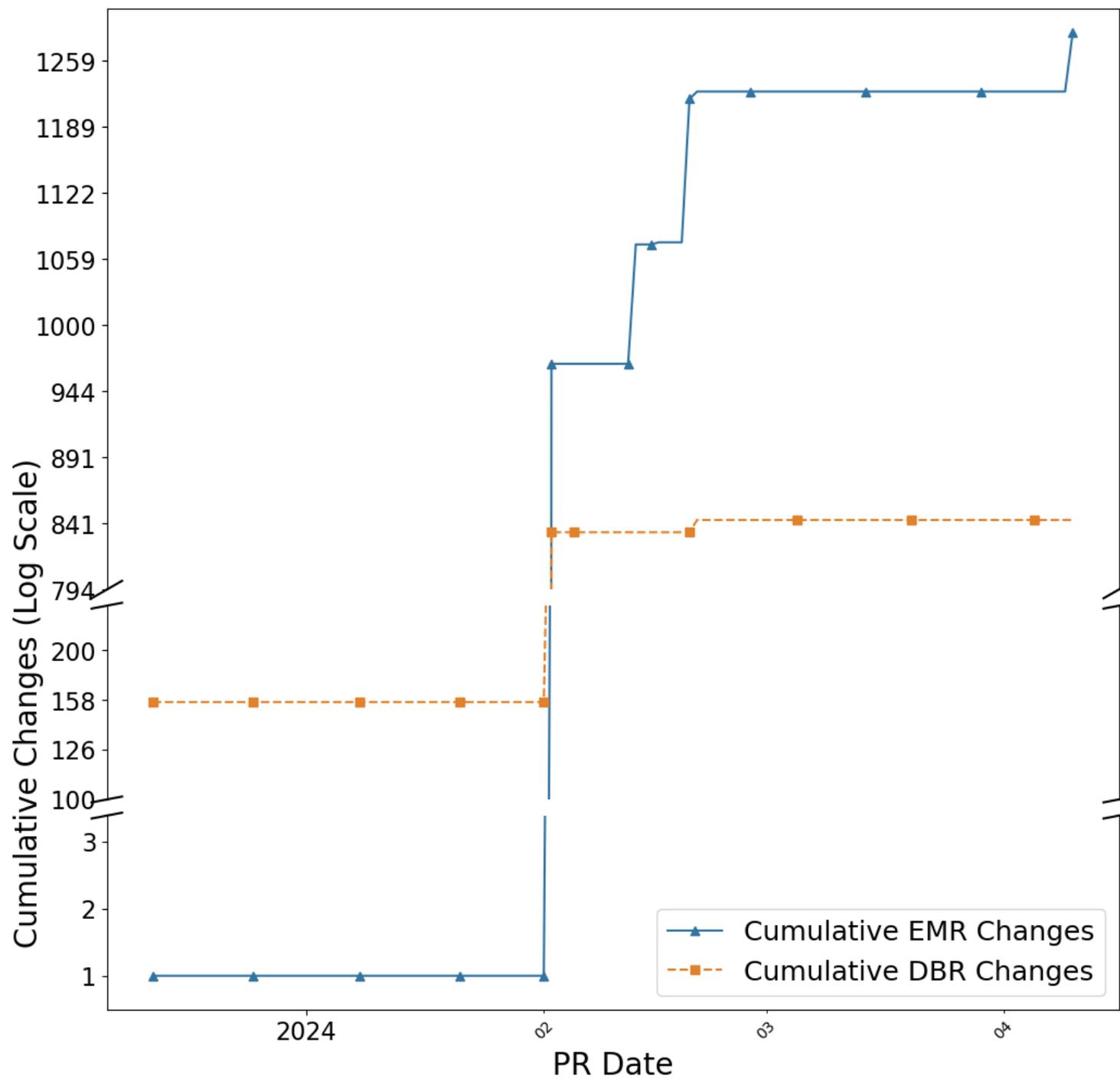
Options

Backfill only failed and missing partitions within selectio

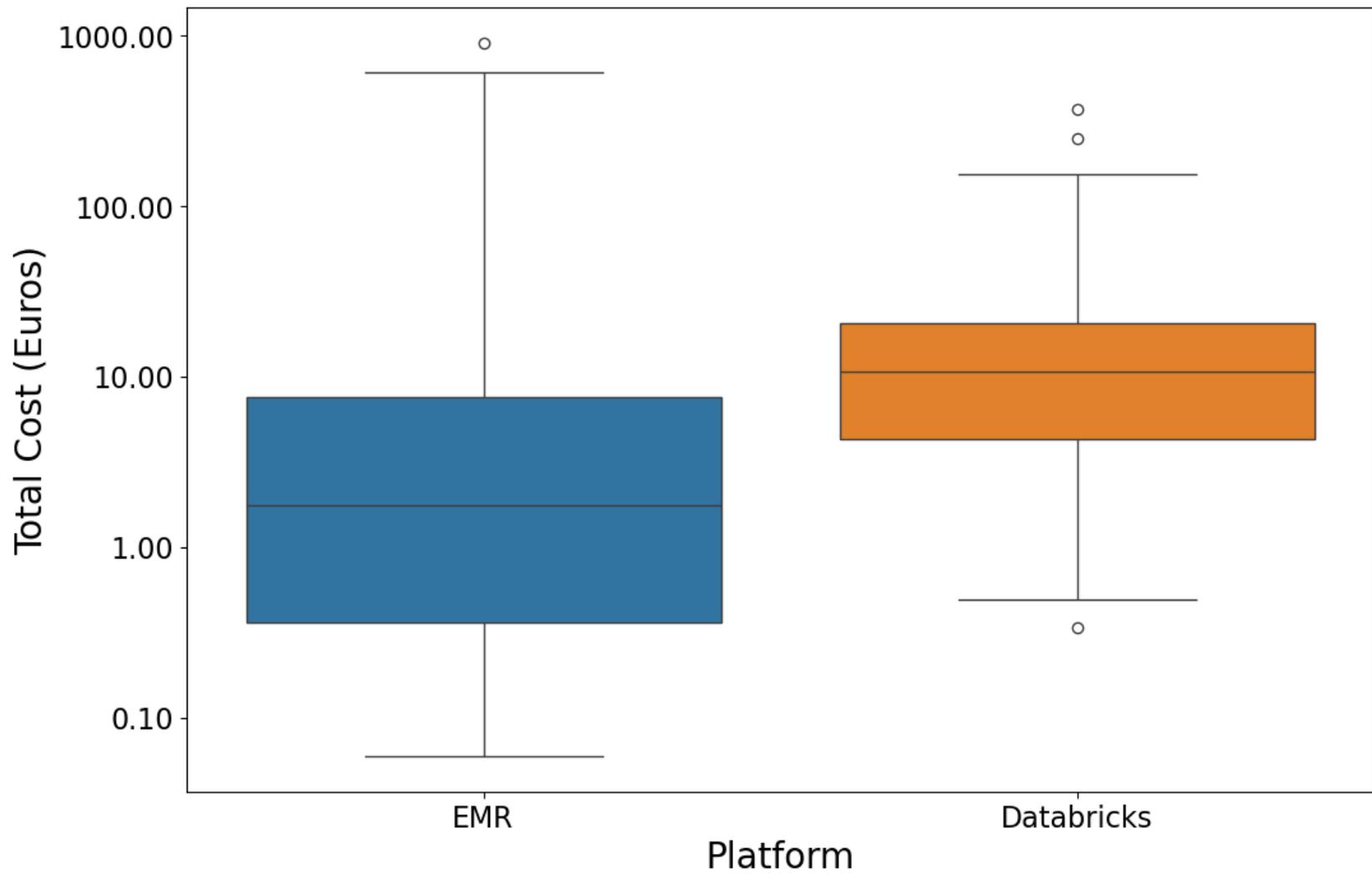
Implementation
time of DBR is
lower



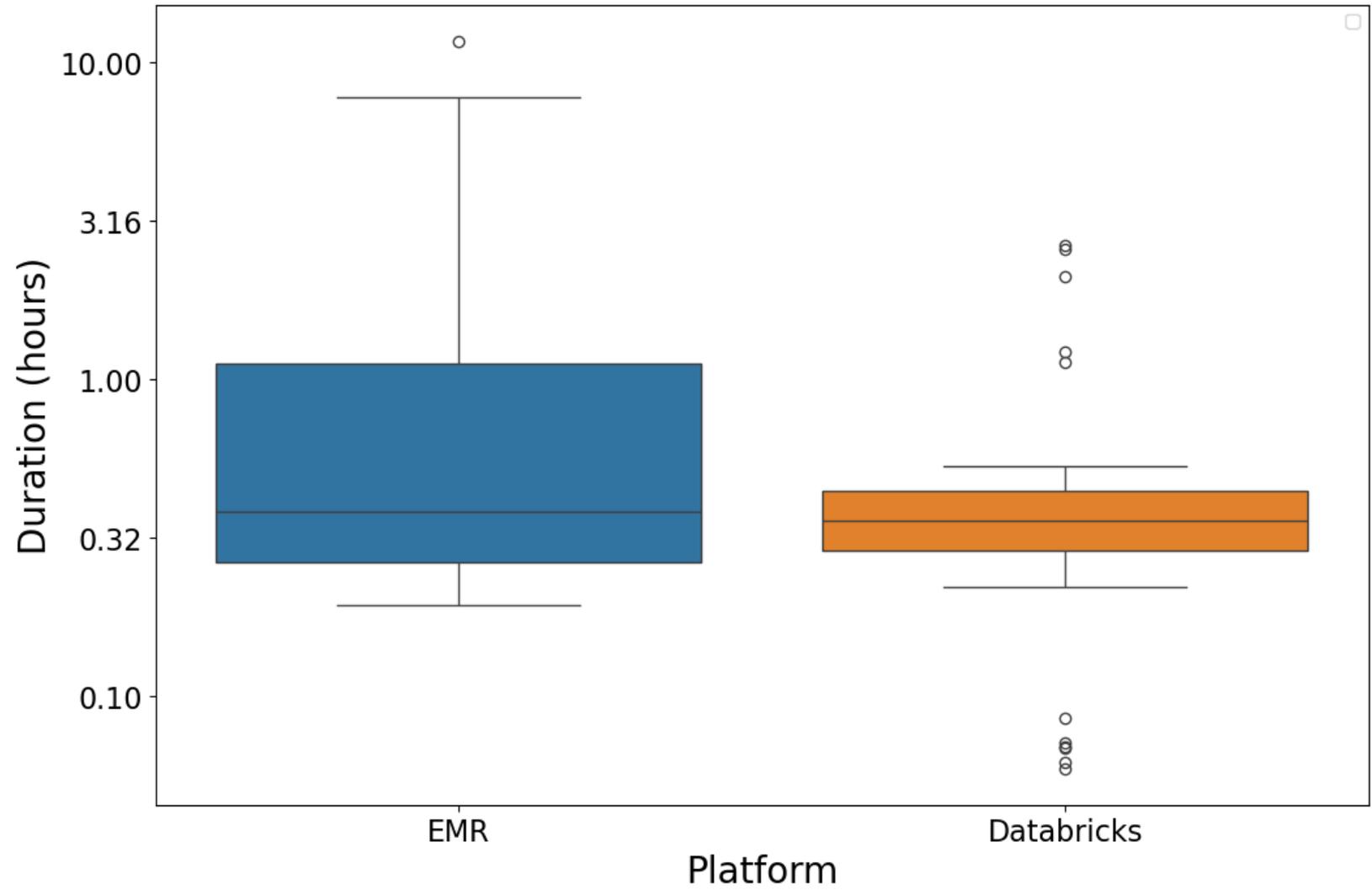
Implementation complexity of DBR is lower more & more frequent commits for EMR integration



Median cost
of DBR is
higher than
EMR



Variability of execution time of DBR is lower



Implementation lessons

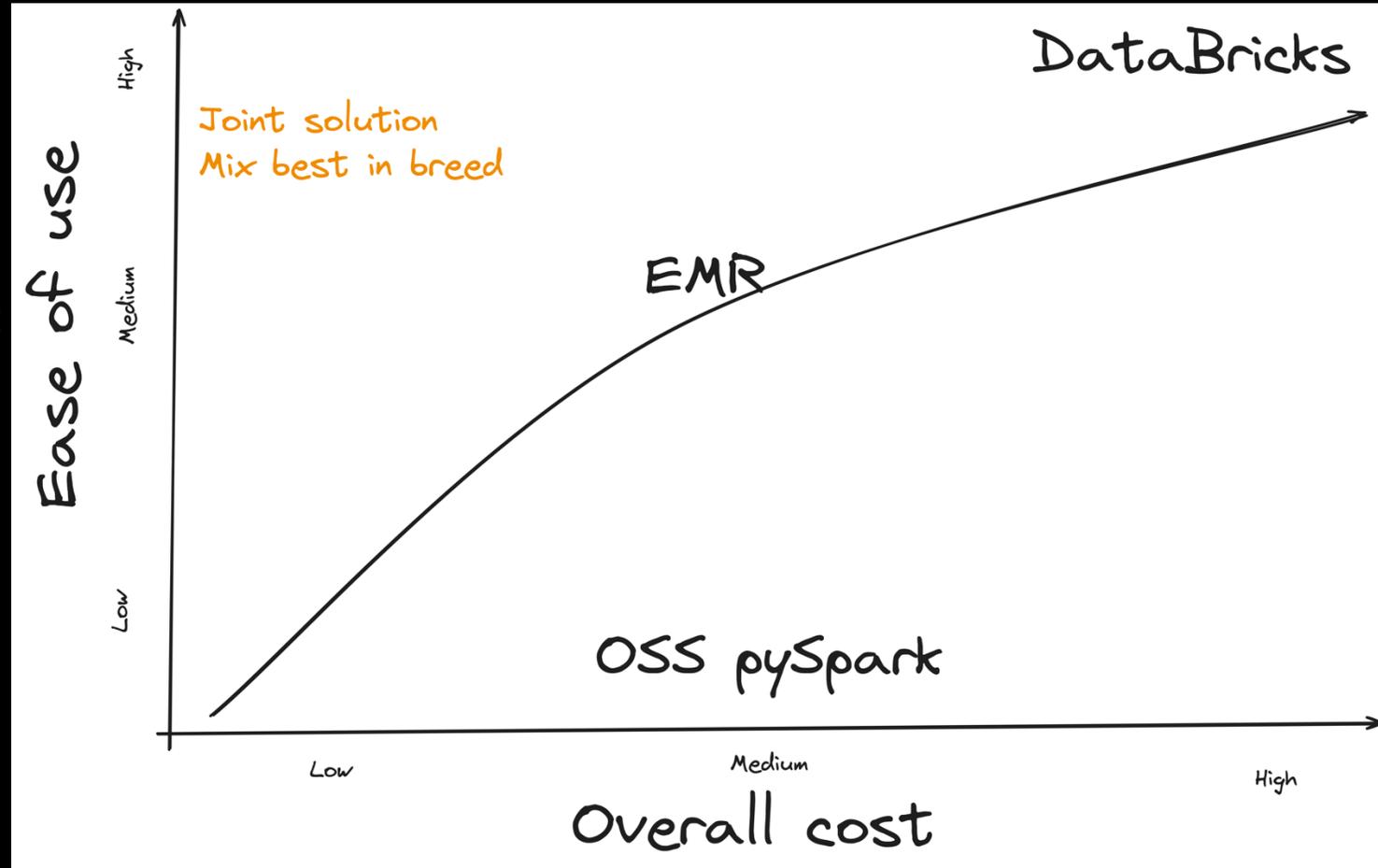
- Complexity of AWS EMR: Many low level details about AWS, spot instances, networking required (master on spot instance => )
- Abstracting the PaaS requires deep understanding of their APIs

Tips

- `maximizeResourceAllocation`
- LZ0
- Delta zorder on partition
- `spark.databricks.delta.vacuum.parallelDelete.enabled=true`

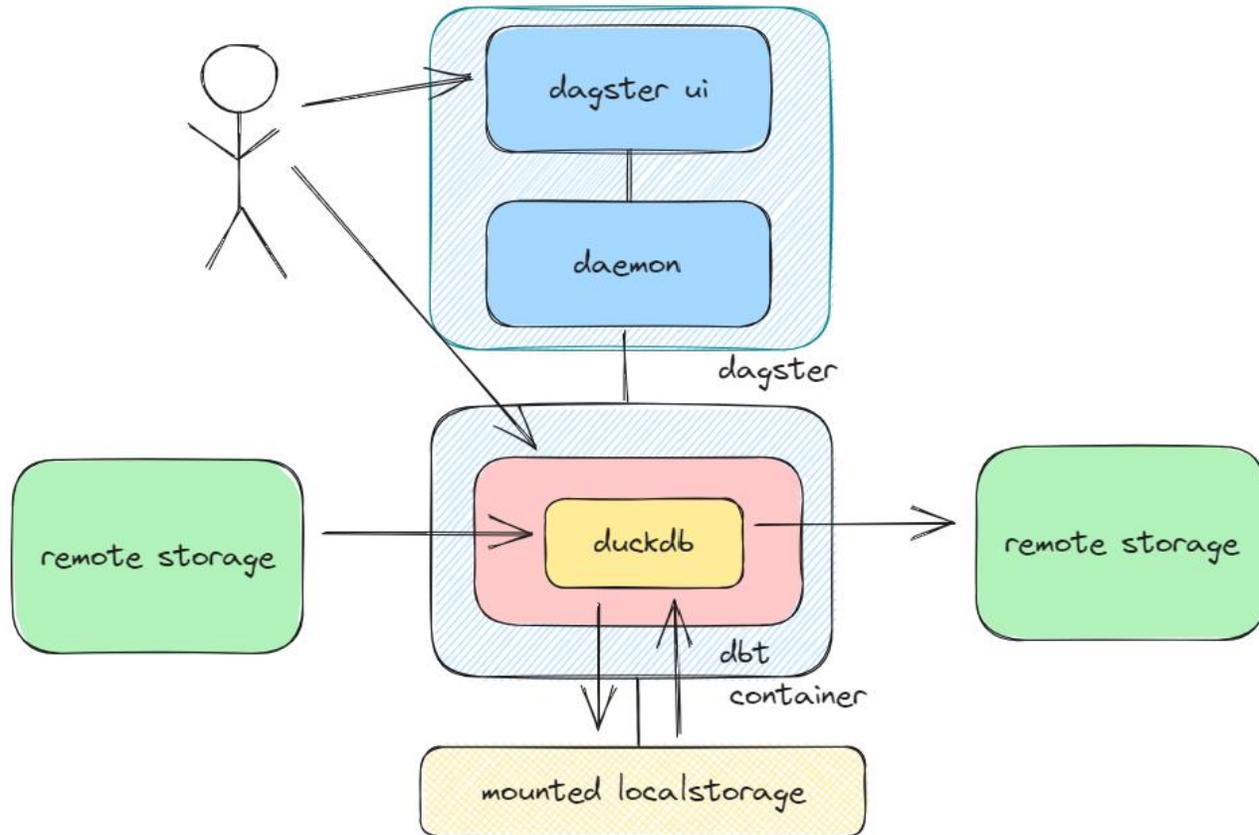
Summary

- Money saved – 43%
- Bring back software engineering best practices for data
- Flexibility
 - Data PaaS as a commodity
 - Take back control
 - Best in breed
- Single pane of glass for pipelines



Takeaway – if you have a small data problem

- Pipes allows to quickly bring in existing scripts whilst retaining observability
- High code engineering practices scales well
- Full control
- Compute technology can easily be changed (i.e. duckdb, daft, ...) data-engineering.expert/2023/12/11/dagster-dbt-duckdb-as-new-local-mps



COST EFFICIENCY FOR DATA

Georg Heiler



bit.ly/efficient-spark

(data-engineering.expert/2024/06/21/cost-efficient-alternative-to-databricks-lock-in
arxiv.org/abs/2408.11635 github.com/ascii-supply-networks/ascii-hydra/tree/main/src/pipelines/ascii_library_demo)