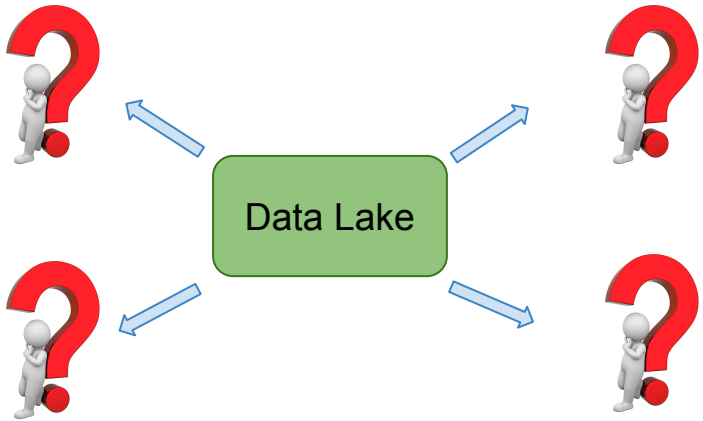
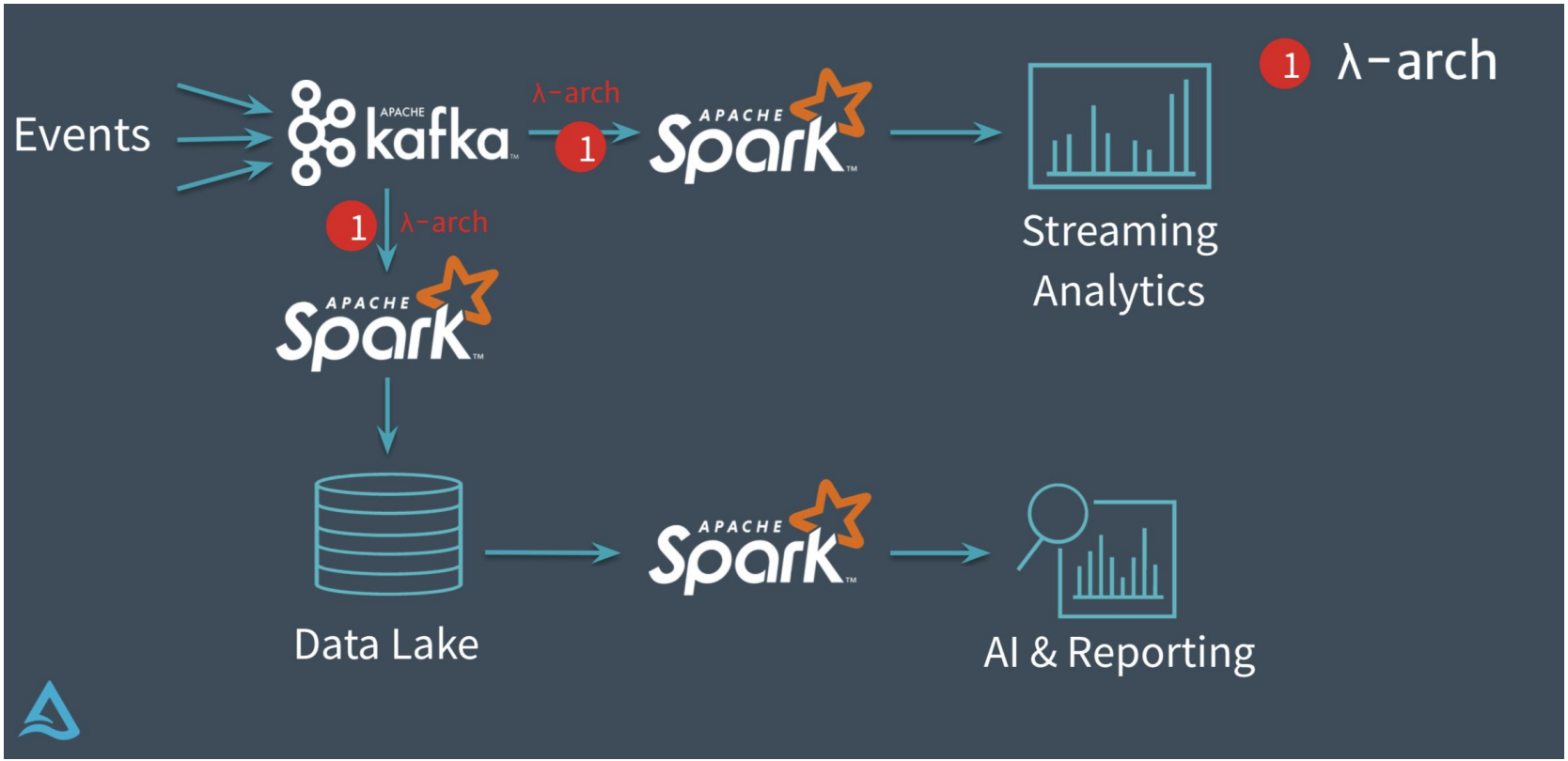
Data Lake现状及选型

openinx@apache.org

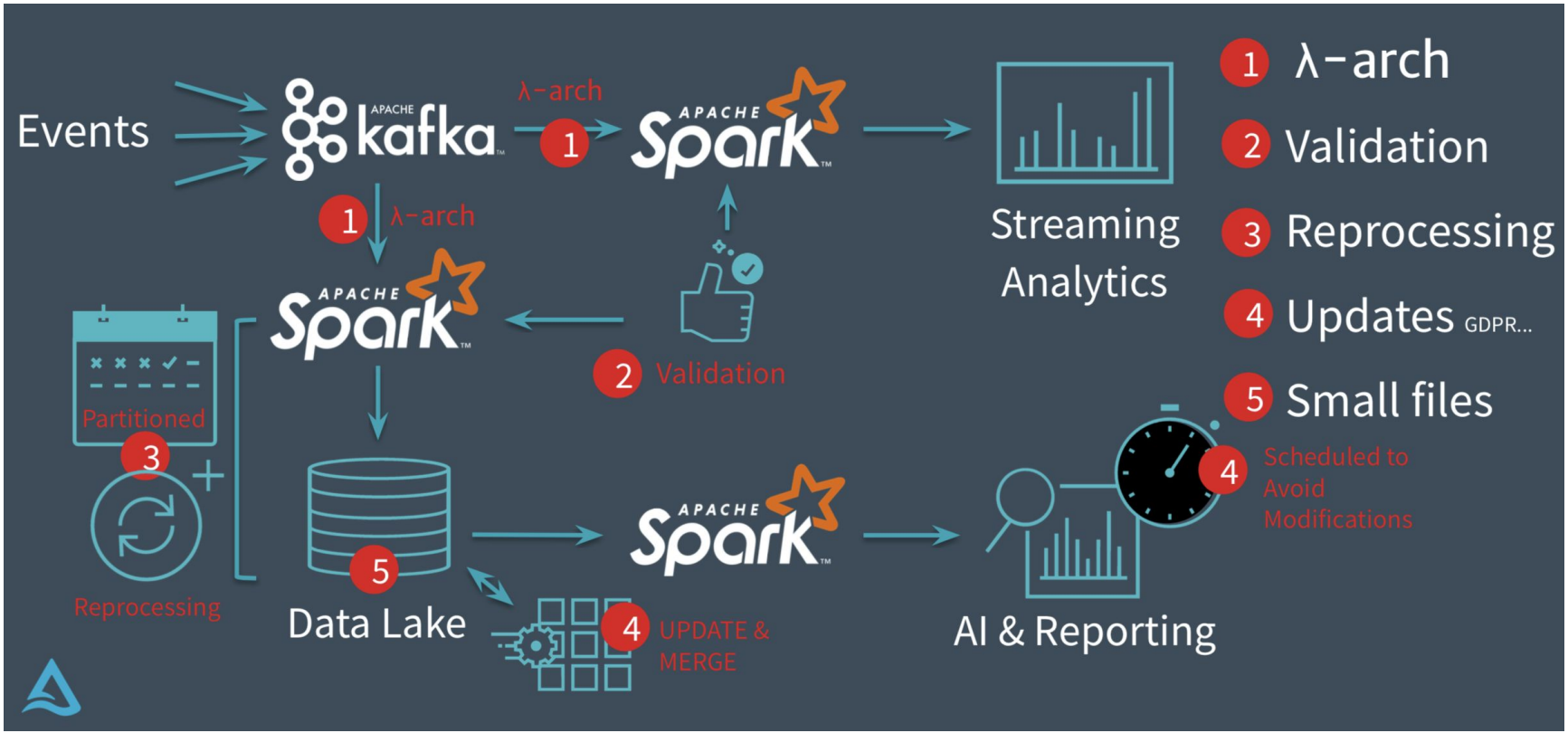
定义数据湖产品



Databricks: Lambda架构



Databricks：存在的问题



Databricks: 现状缺陷

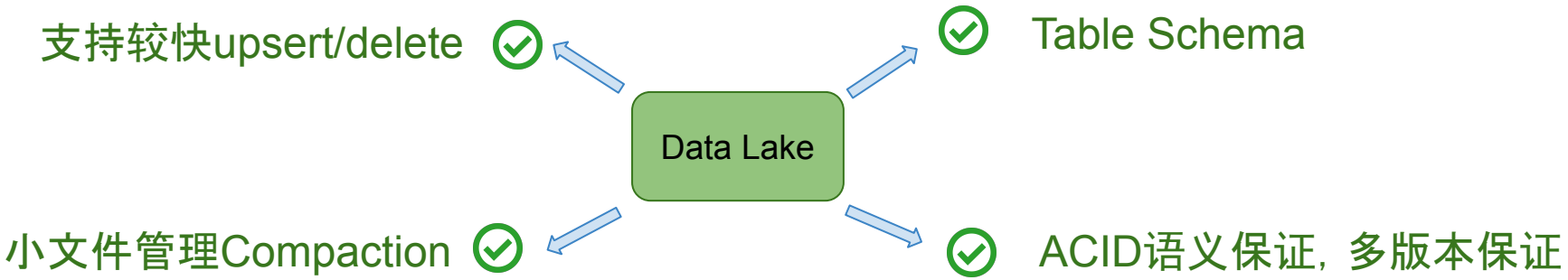
- 缺乏Table Schema，数据质量没法保证
  - 导入数据的数据类型不确定
  - 字段数无法确定
  - 数据格式不固定
- 无法保证ACID语义
  - 可能写入一部分数据到FileSystem
  - 可能读取到导入一半的数据
  - 无法追溯完整历史版本
- Upsert成本高
  - GDPR策略必须要求数据物理删除
- 可能造成很多小文件
  - 文件系统管理原数据成本高
  - 做directory list操作慢

Databricks: 现状缺陷

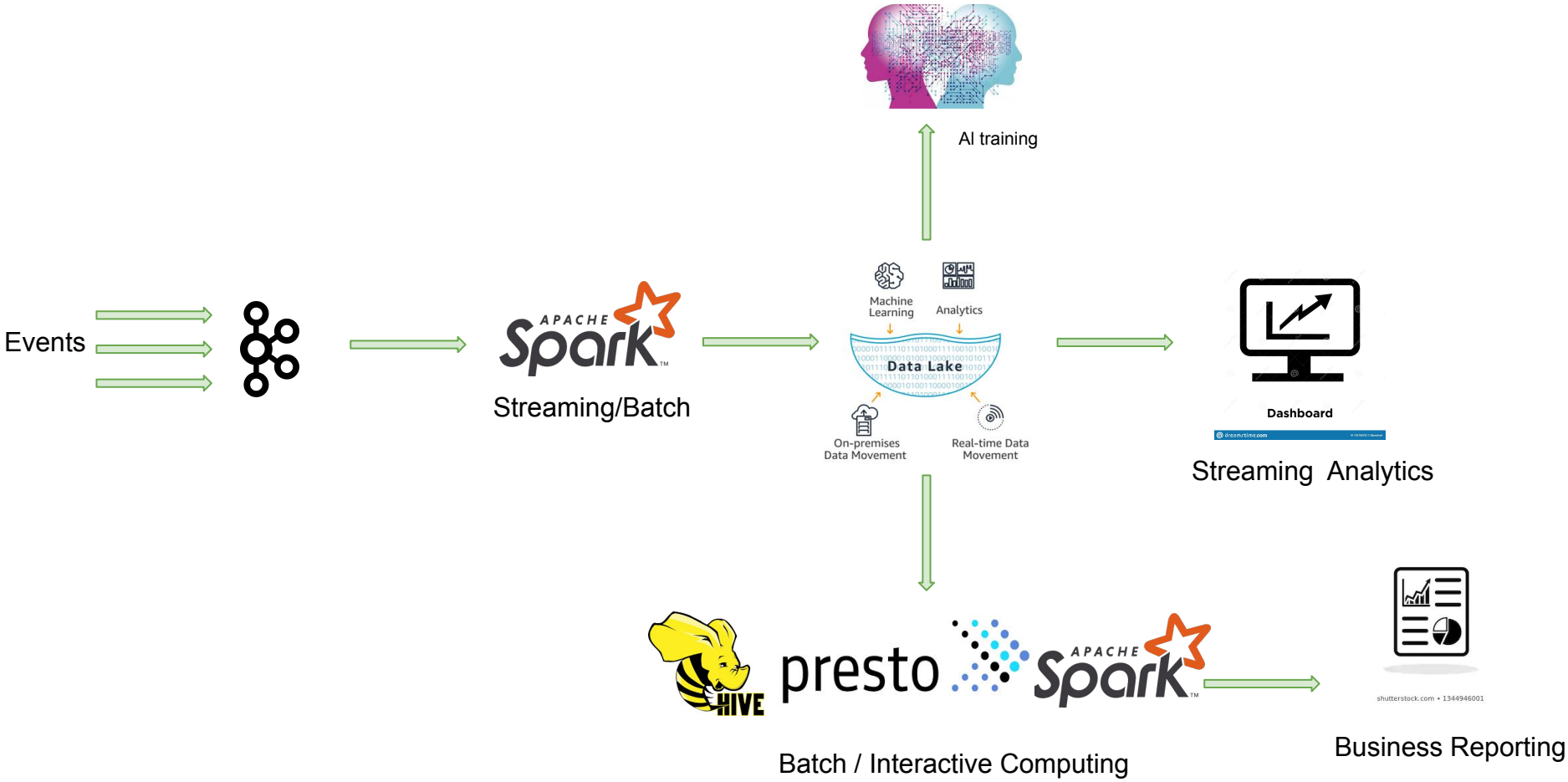
- 缺乏Table Schema，数据质量没法保证
  - 导入数据的数据类型不确定
  - 字段数无法确定
  - 数据格式不固定
- 无法保证ACID语义
  - 可能写入一部分数据到 FileSystem
  - 可能读取到导入一半的数据
  - 无法追溯完整历史版本
- Upsert成本高
  - GDPR策略必须要求数据物理删除
- 可能造成很多小文件

- 强Table Schema
- ACID语义保证，历史版本保证
- 较快速的upsert/delete
- 小文件管理机制

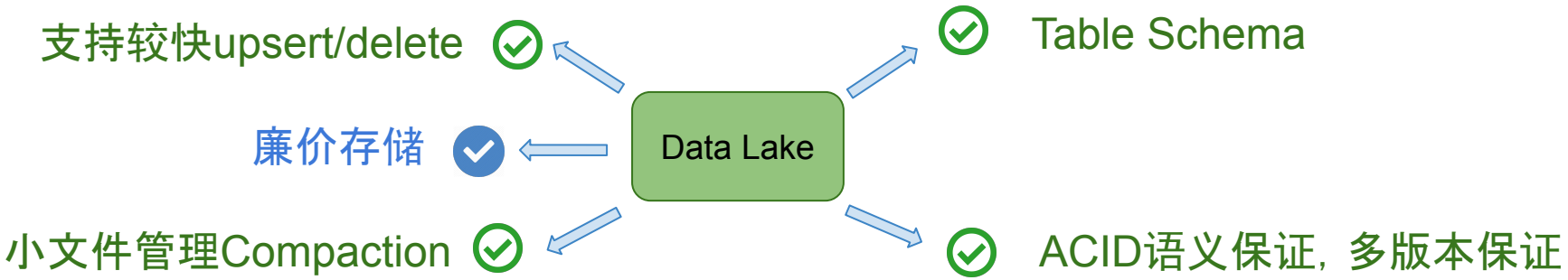
Databricks 数据湖场景总结



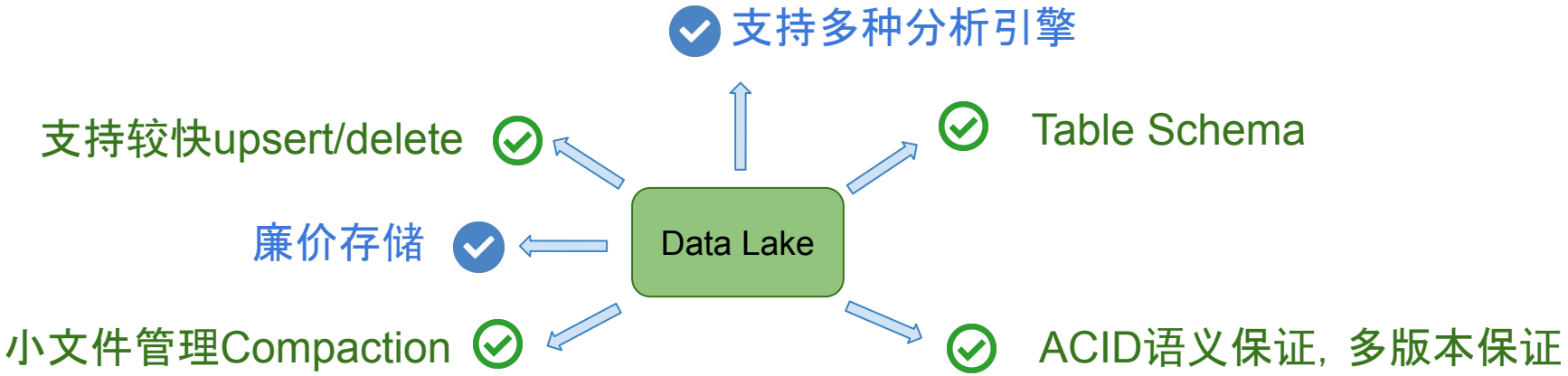
Databricks: Delta数据湖



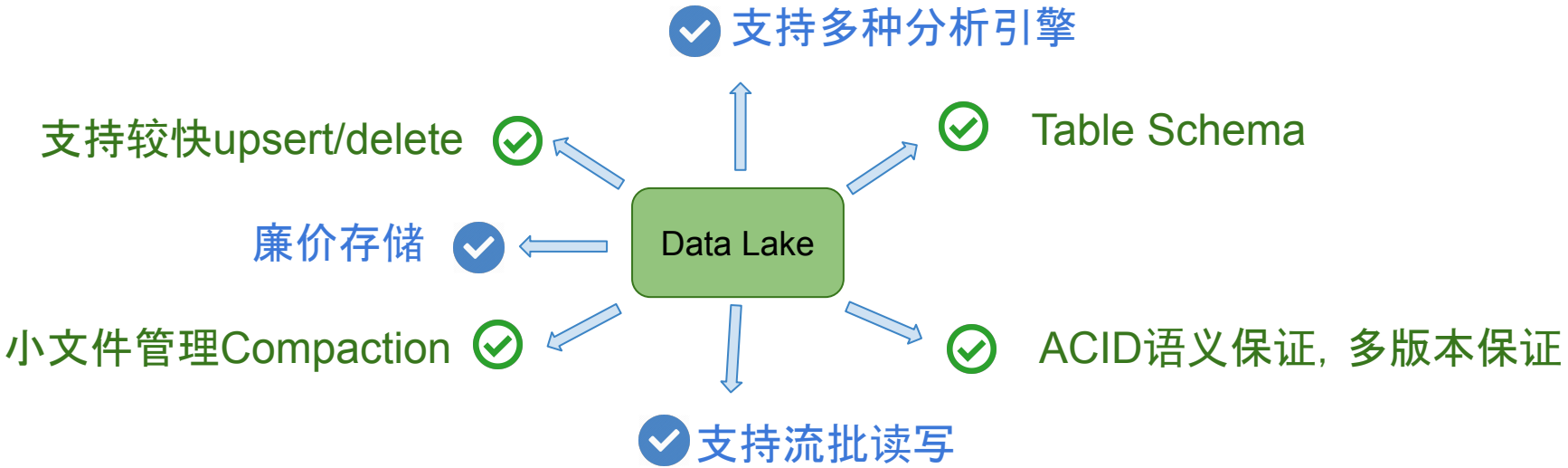
Databricks数据湖场景总结



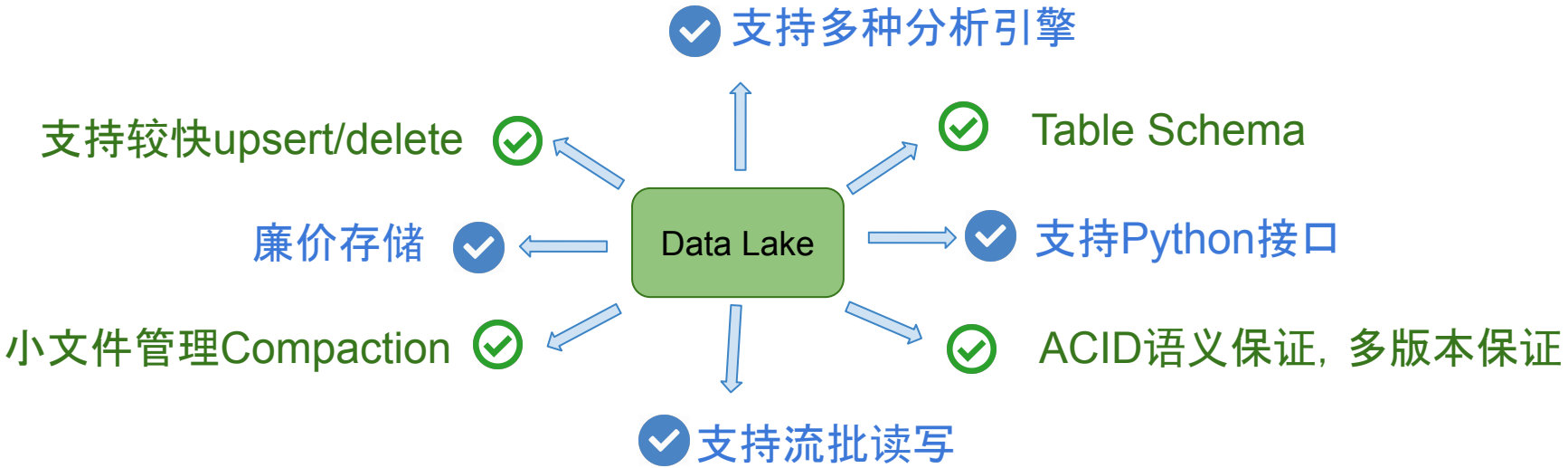
Databricks数据湖场景总结



Databricks数据湖场景总结

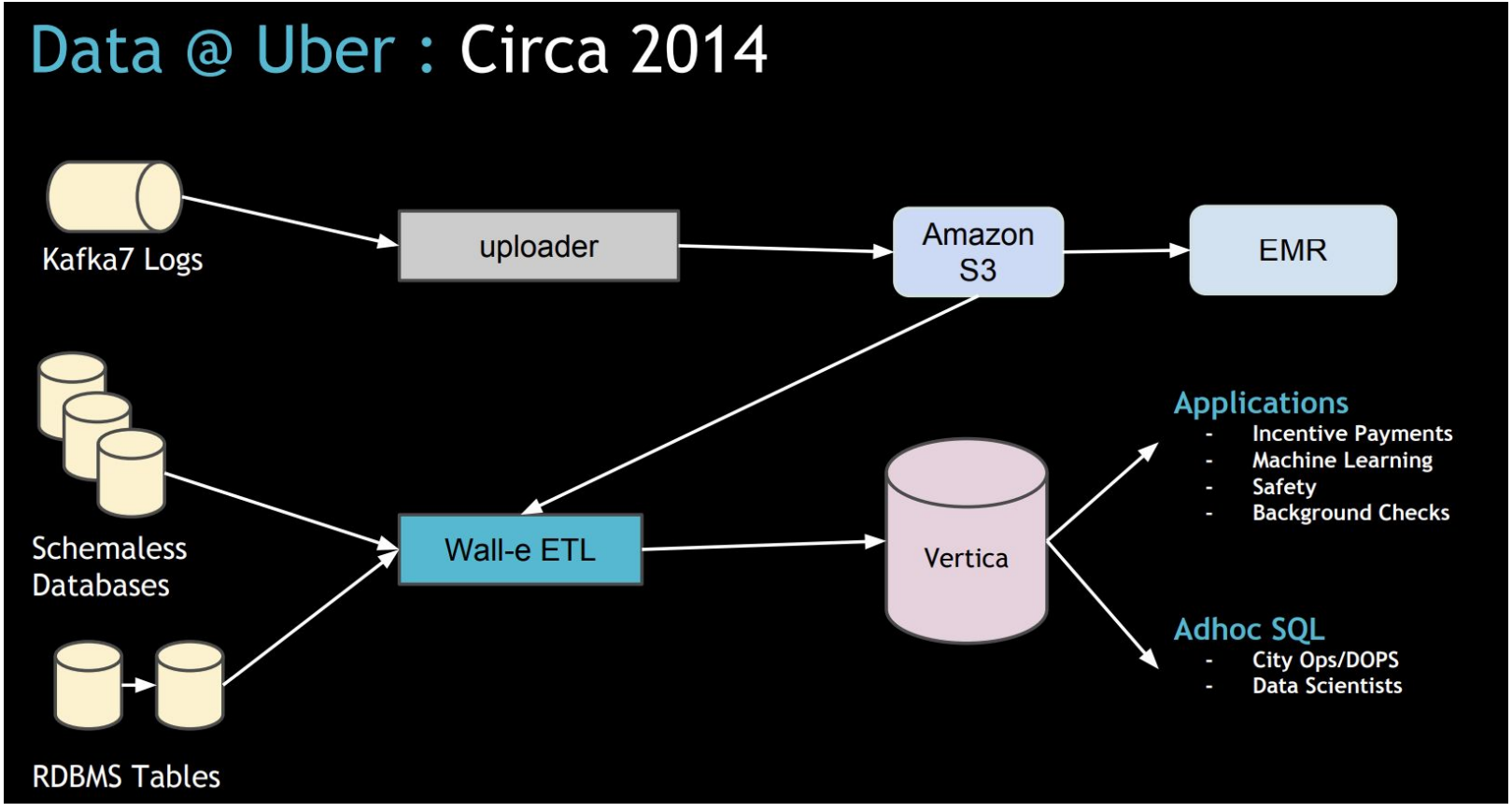


Databricks数据湖场景总结

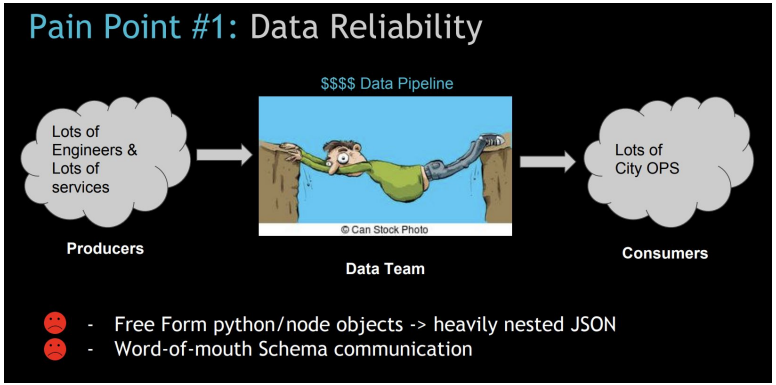


- **数据湖用户场景分析**
  - Databricks：Lambda架构问题及改进 (link)
  - **Uber: 城市运营人员要求更实时的行程数据** (link)
  - Netflix：Hive的痛点 (link)
- Flink数据湖产品技术选型
  - 8个维度深度对比Delta、Hudi、Iceberg
  - 我们的选型是 ？
- 其他
  - Kafka + KSQL
  - AWS数据湖现状

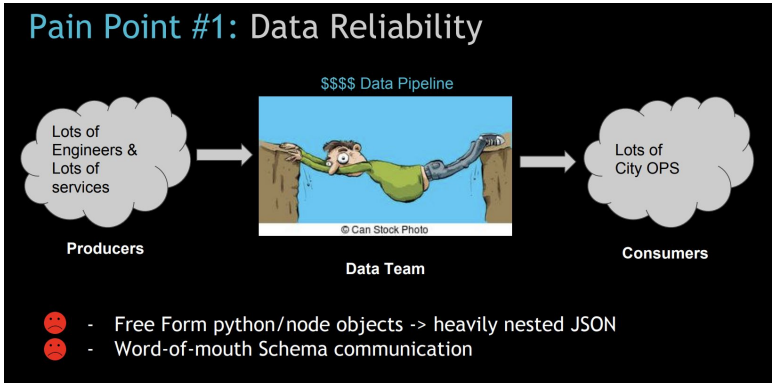
Uber: 2014年数据线

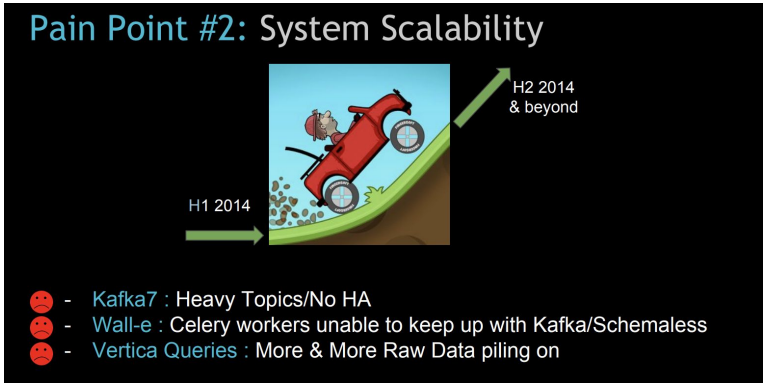


Uber: 存在问题


Pain Point #1: Data Reliability
$$$$ Data Pipeline
Lots of Engineers & Lots of services
Producers
© Can Stock Photo
Data Team
Lots of City OPS
Consumers
- Free Form python/node objects -> heavily nested JSON
- Word-of-mouth Schema communication

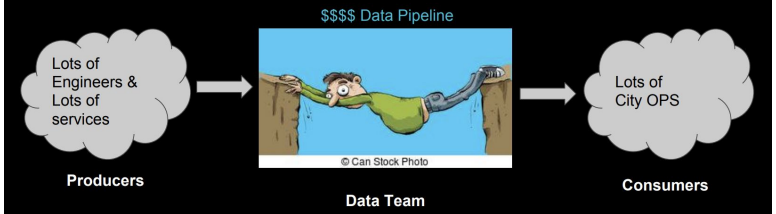
Uber: 存在问题


Pain Point #1: Data Reliability
$$$$ Data Pipeline
Lots of Engineers & Lots of services
Producers
© Can Stock Photo
Data Team
Lots of City OPS
Consumers
- Free Form python/node objects -> heavily nested JSON
- Word-of-mouth Schema communication


Pain Point #2: System Scalability
H1 2014
H2 2014
& beyond
- Kafka7 : Heavy Topics/No HA
- Wall-e : Celery workers unable to keep up with Kafka/Schemaless
- Vertica Queries : More & More Raw Data piling on

Uber: 存在问题

Pain Point #1: Data Reliability

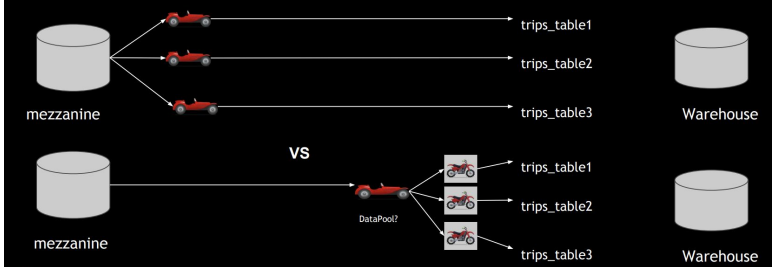


- Free Form python/node objects -> heavily nested JSON
- Word-of-mouth Schema communication

Pain Point #2: System Scalability

- Kafka7 : Heavy Topics/No HA
- Wall-e : Celery workers unable to keep up with Kafka/Schemaless
- Vertica Queries : More & More Raw Data piling on

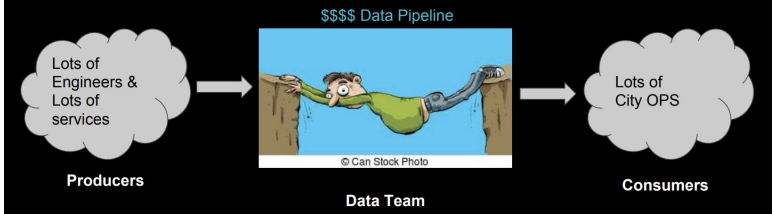
Pain Point #3: Fragile Ingestion Model



- Multiple fetching from sources
- Painful Backfills, since projections & transformation are in pipelines

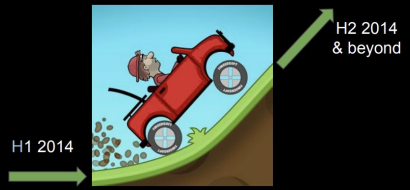
Uber: 存在问题

Pain Point #1: Data Reliability



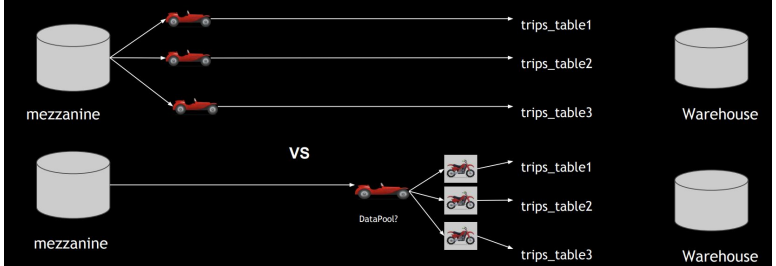
- Free Form python/node objects -> heavily nested JSON
- Word-of-mouth Schema communication

Pain Point #2: System Scalability



- Kafka7 : Heavy Topics/No HA
- Wall-e : Celery workers unable to keep up with Kafka/Schemaless
- Vertica Queries : More & More Raw Data piling on

Pain Point #3: Fragile Ingestion Model

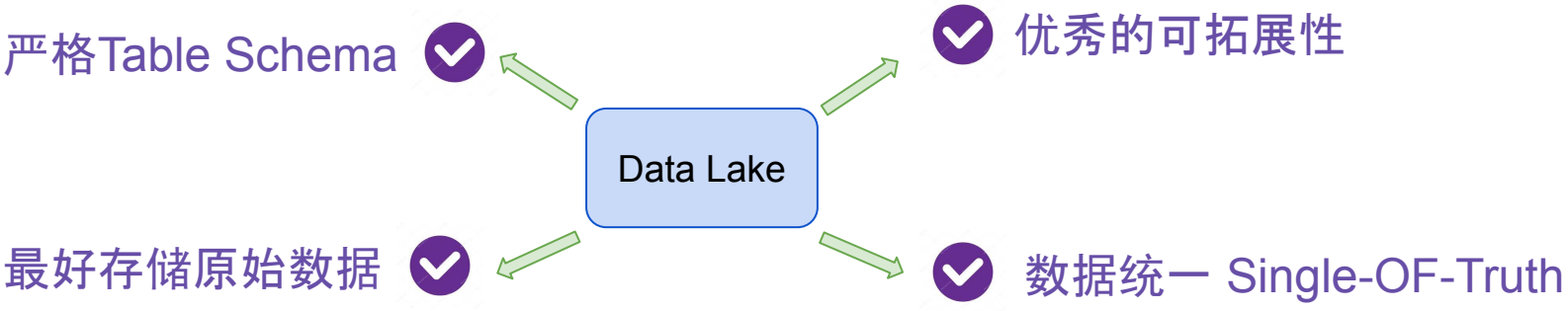


- Multiple fetching from sources
- Painful Backfills, since projections & transformation are in pipelines

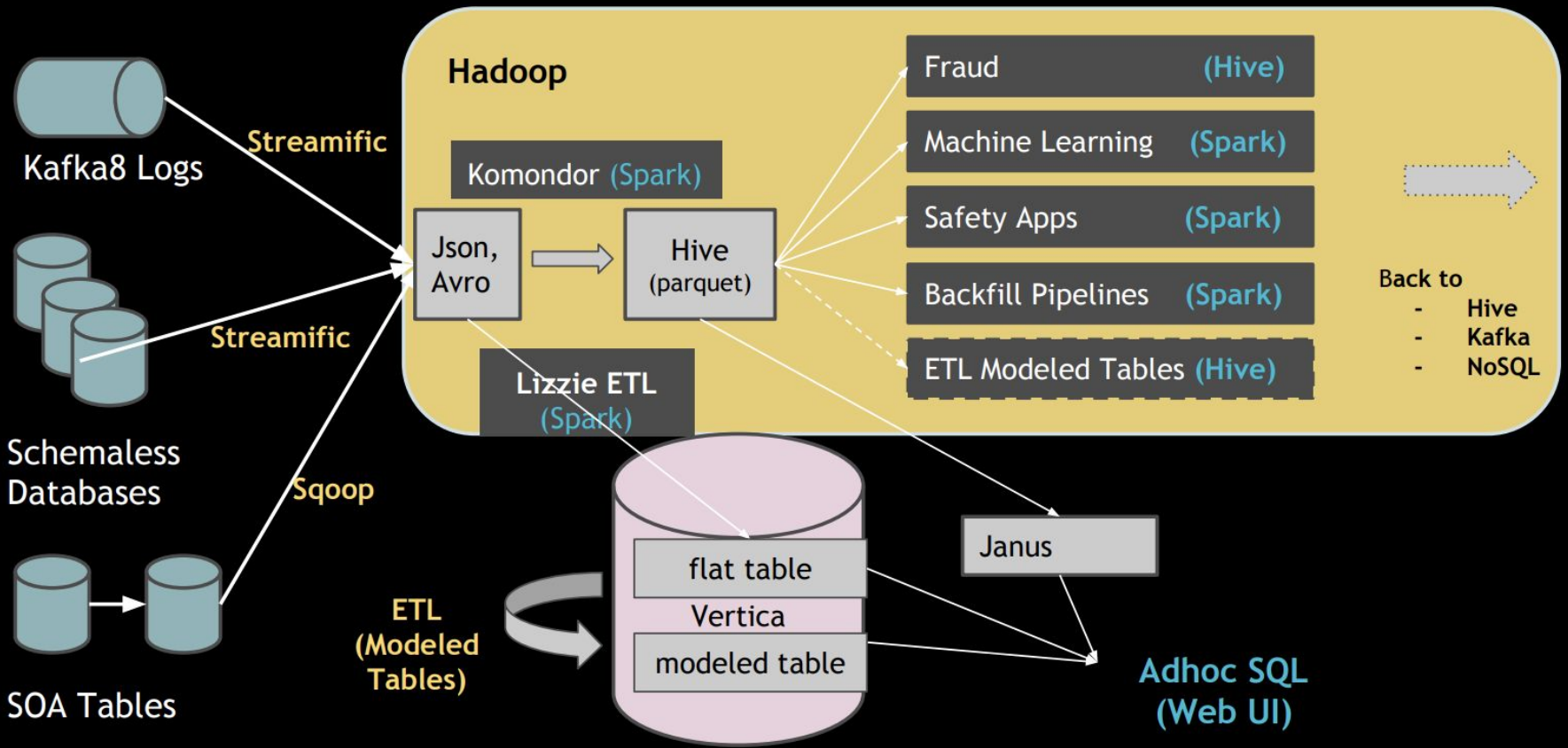
Pain Point #4: No Multi-DC Support

- No Unified view of data, More complexity from consumer
- Wasteful use of WAN traffic

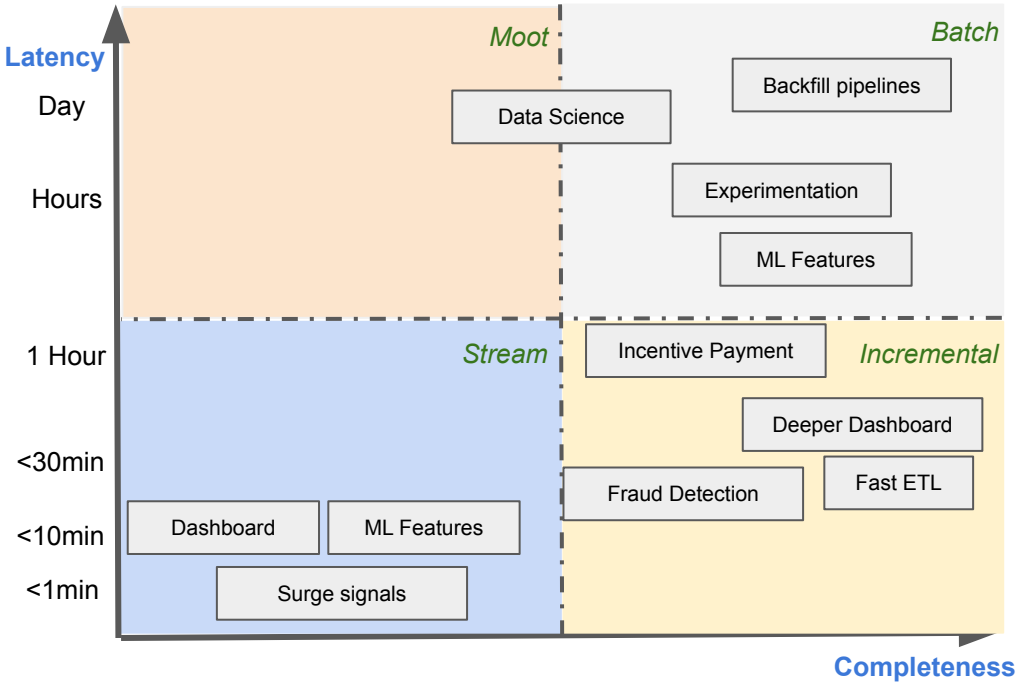
Uber数据湖场景总结



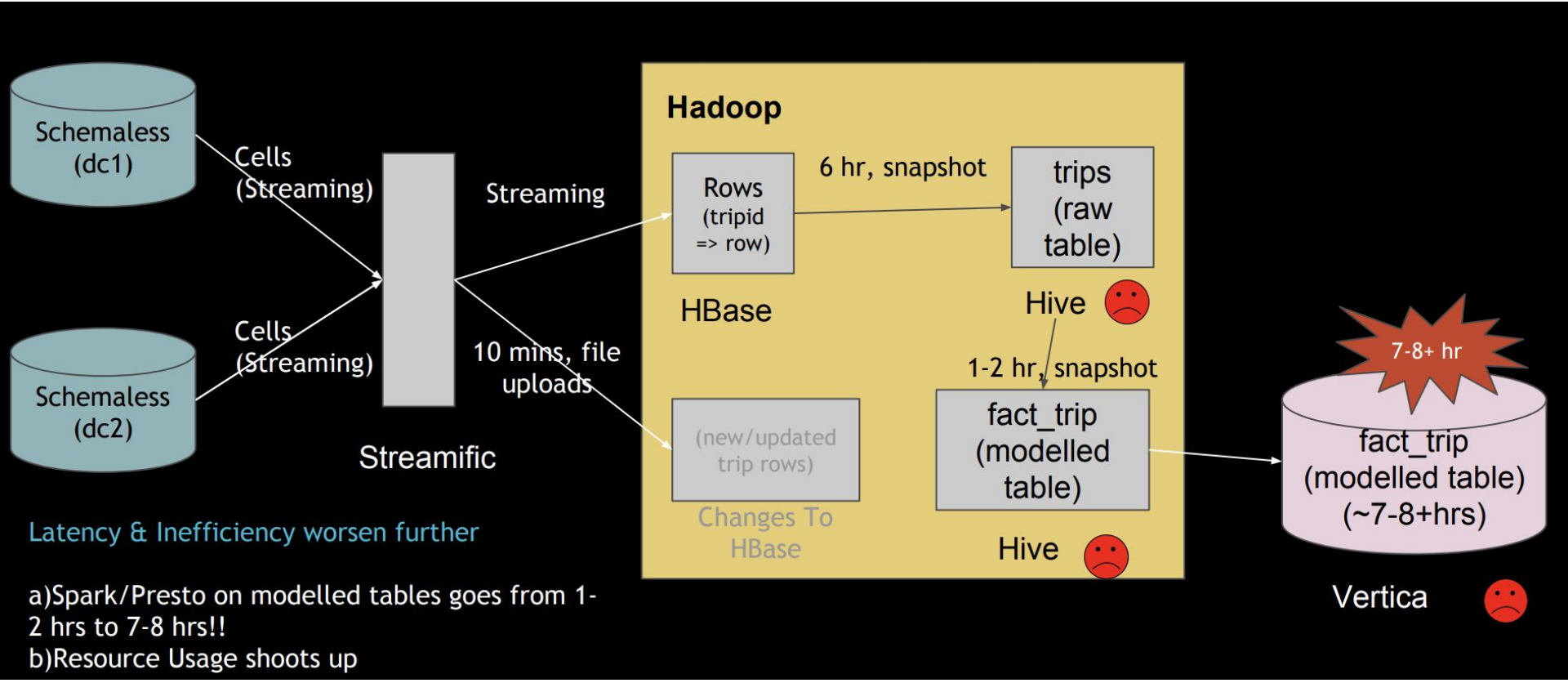
Hadoop Ecosystem: Overview



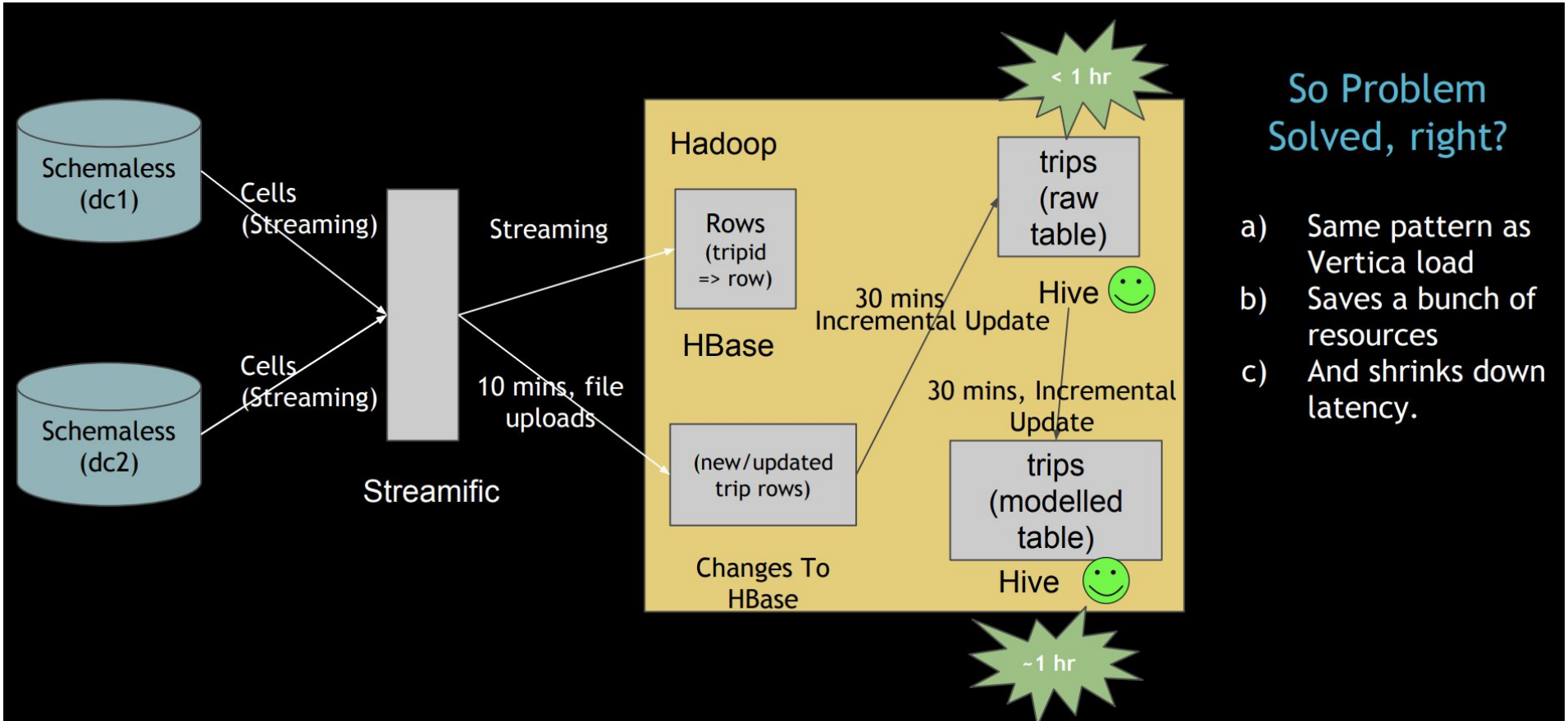
Uber: Incremental Model



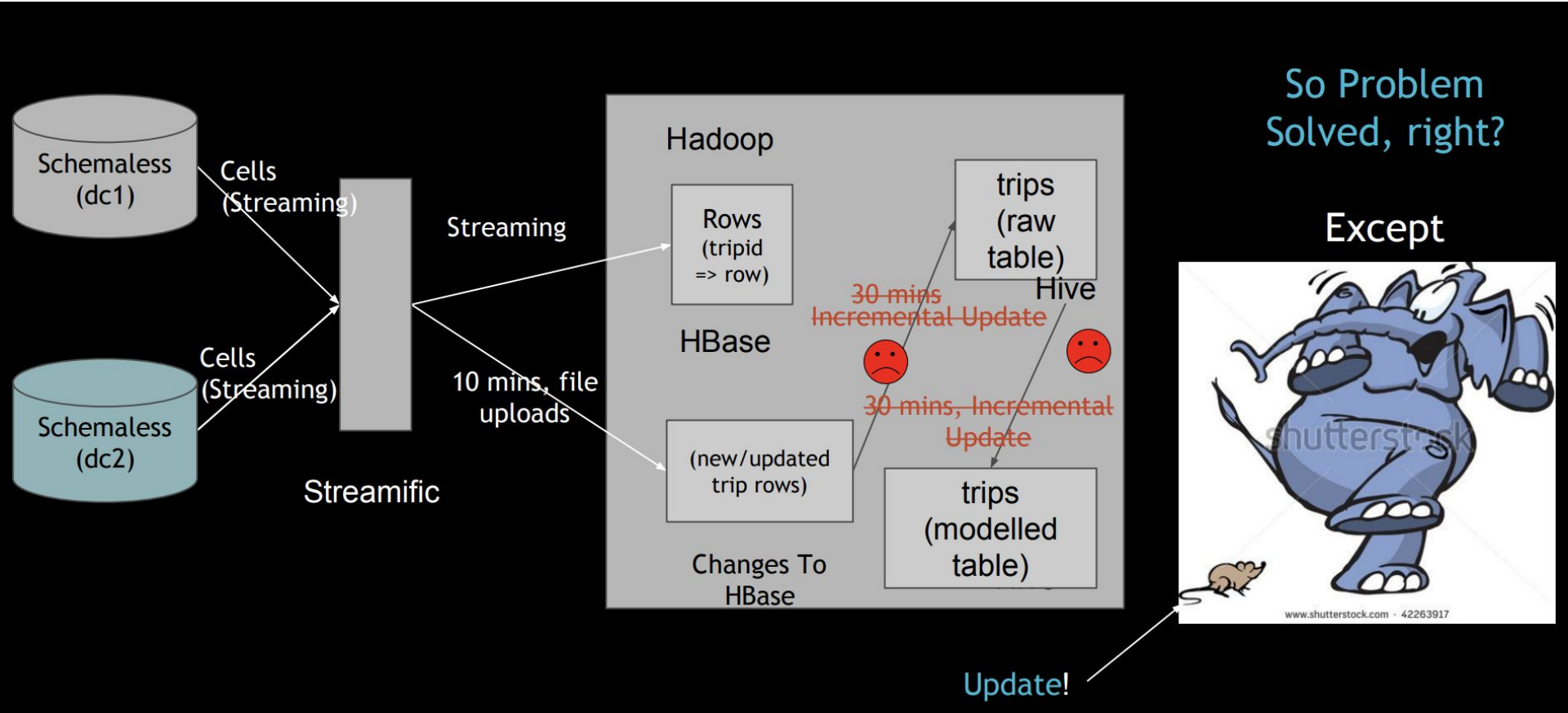
Uber: 数据延迟太大？


Schemaless (dc1)
Schemaless (dc2)
Cells (Streaming)
Cells (Streaming)
Streamific
Streaming
10 mins, file uploads
Hadoop
Rows (tripid => row)
HBase
6 hr, snapshot
trips (raw table)
Hive
1-2 hr, snapshot
(new/updated trip rows)
Changes To HBase
fact_trip (modelled table)
Hive
7-8+ hr
fact_trip (modelled table) (~7-8+hrs)
Vertica
Latency & Inefficiency worsen further
a)Spark/Presto on modelled tables goes from 1-2 hrs to 7-8 hrs!!
b)Resource Usage shoots up

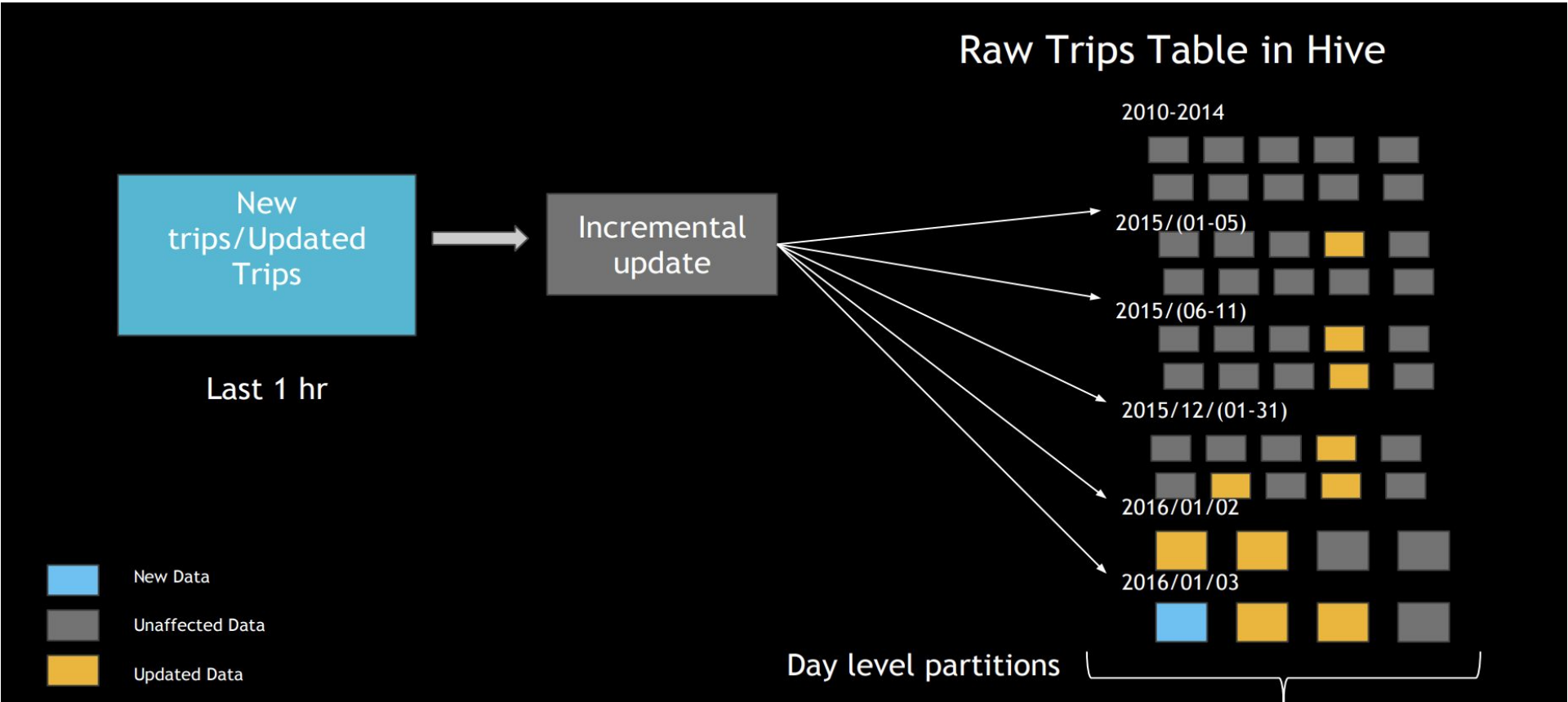
Uber: 增量导入数据是否可行



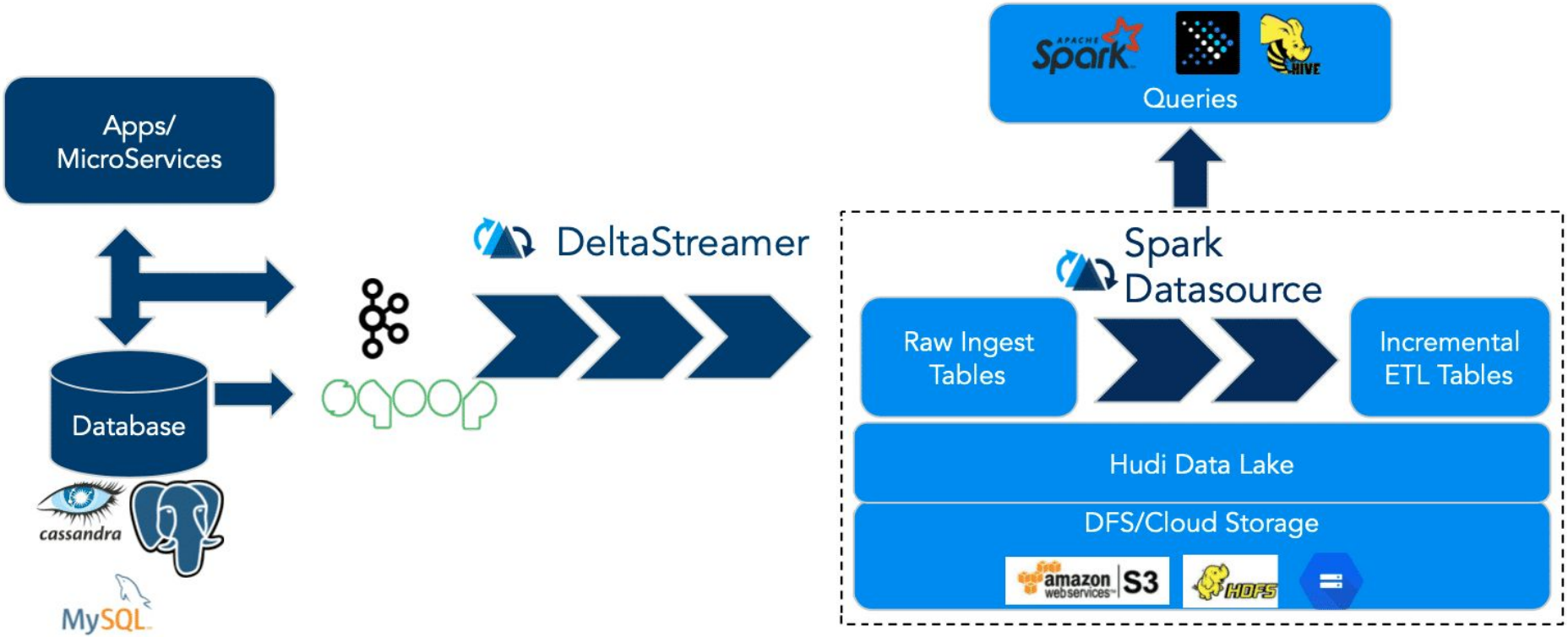
Uber: Update是增量更新的障碍？



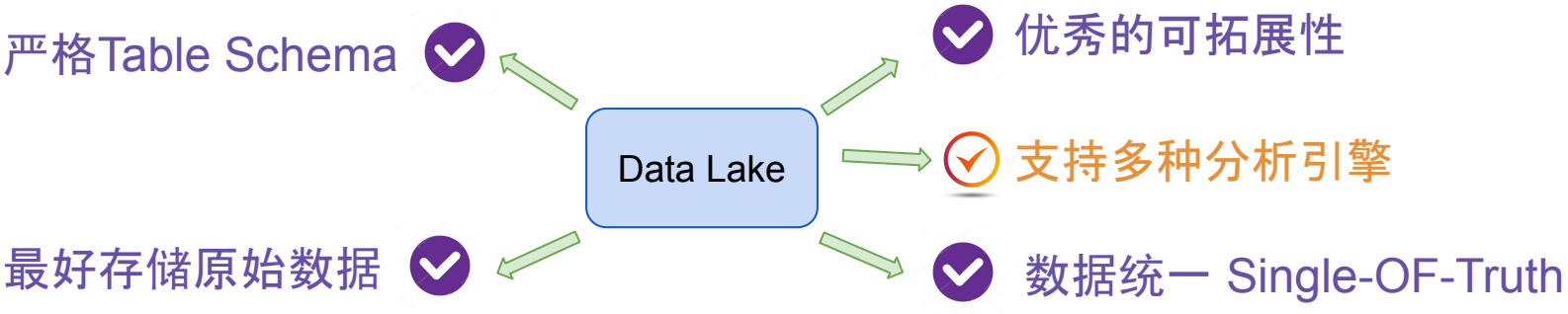
Uber: 快速增量update



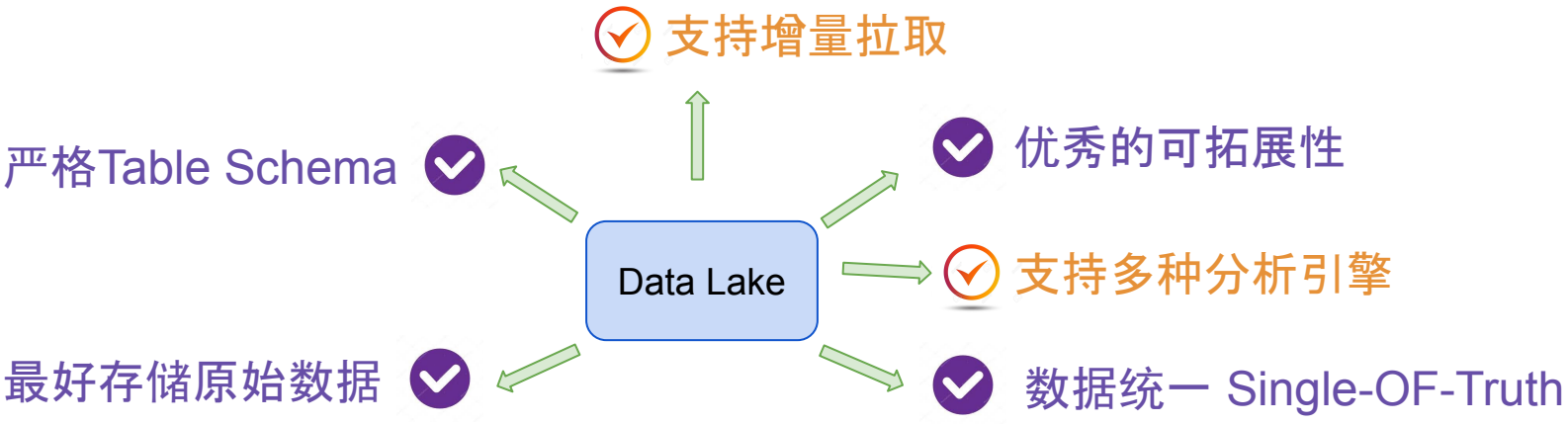
Uber: 选择Hudi


Apps/
MicroServices
Database
cassandra
MySQL
DeltaStreamer
Queries
Spark
Datasource
Raw Ingest
Tables
Incremental
ETL Tables
Hudi Data Lake
DFS/Cloud Storage
amazon
web services
S3
HDFS

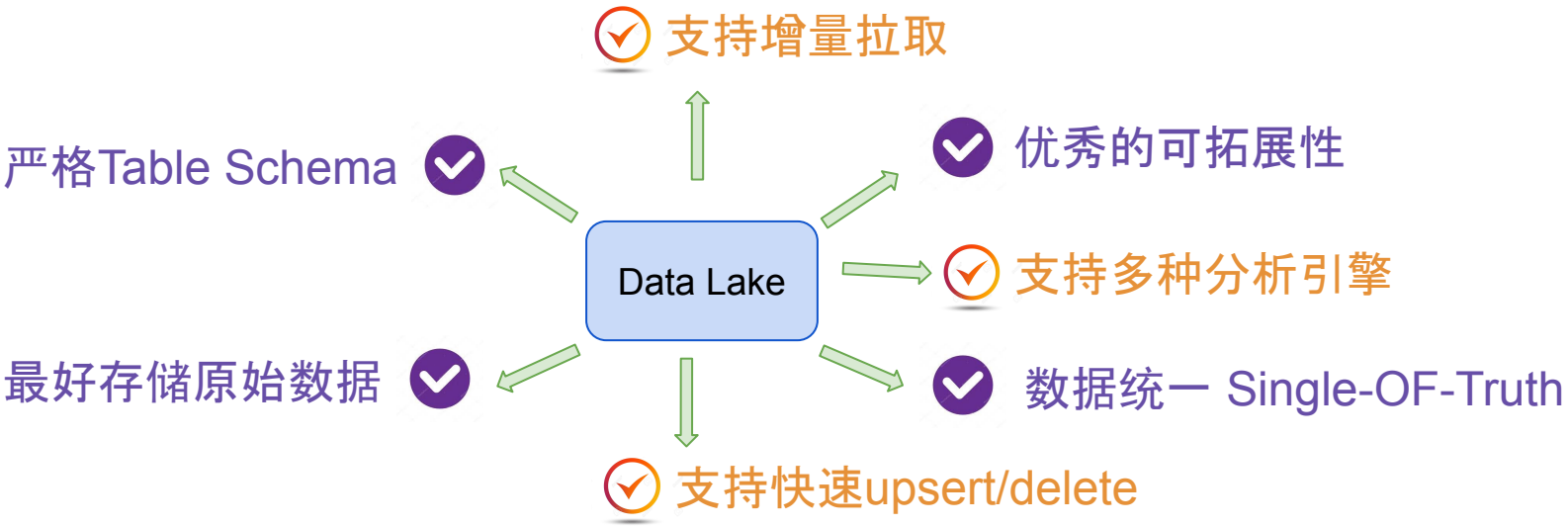
Uber数据湖场景总结



Uber数据湖场景总结



Uber数据湖场景总结



- **数据湖用户场景分析**
  - Databricks: Lambda架构问题及改进 (link)
  - Uber: 城市运营人员要求更实时的行程数据 (link)
  - **Netflix: Hive的痛点** (link)
- 数据湖产品对比
  - 8个维度深度对比Delta、Hudi、Iceberg
  - 选型 ？
- 其他
  - Kafka + KSQL
  - AWS数据湖现状

Netflix: Hive的痛点

Case Study: Netflix Atlas

- Historical Atlas data:
  - Time-series metrics from Netflix runtime systems
  - 1 month: 2.7 million files in 2,688 partitions
  - Problem: cannot process more than a few days of data

- Sample query:

```
select distinct tags['type'] as type
from iceberg.atlas
where
  name = 'metric-name' and
  date > 20180222 and date <= 20180228
order by type;
```

Netflix: Hive的痛点

Atlas Historical Queries

- Hive table – with Parquet filters:
  - 400k+ splits, not combined
  - EXPLAIN query: **9.6 min** (planning wall time)

Netflix: Hive的痛点

Hive Metastore

- HMS keeps metadata in SQL database
  - Tracks information about partitions
  - Tracks schema information
  - Tracks table statistics
- Allows filtering by partition values
  - Filters only pushed to DB for string types
- Uses external SQL database
  - Metastore is often the bottleneck for query planning
- Only file system tracks the files in each partition…
  - No per-file statistics

Netflix: Hive的痛点

Hive Metastore

- HMS keeps metadata in SQL database
  - Tracks information about partitions
  - Tracks schema information
  - Tracks table statistics
- Allows filtering by partition values
  - Filters only pushed to DB for string types
- Uses external SQL database
  - Metastore is often the bottleneck for query planning
- Only file system tracks the files in each partition...
  - No per-file statistics

🚫 SQL-DB信息太多，性能差

🚫 非Partition字段filter，性能差

🚫 集中式外部SQL-DB，性能差

🚫 无文件级别统计信息，性能差

Netflix: Hive的痛点

Hive ACID layout

- Provides snapshot isolation and atomic updates
- Transaction state is stored in the metastore
- Uses the same partition/directory layout
  - Creates new directory structure inside partitions

```
date=20180513/
 |- hour=19/
 |    |- base_0000000/
 |    |    |- bucket_00000
 |    |    |- ...
 |    |    |- bucket_00031
 |    |- delta_0000001_0000100/
 |    |    |- bucket_00000
 |    |    |- ...
```

Netflix: Hive的痛点

Design Problems

- Table state is stored in two places
  - Partitions in the Hive Metastore
  - Files in a file system
- Bucketing is defined by Hive's (Java) hash implementation.
- Non-ACID layout's only atomic operation is add partition
- **Requires atomic move of objects in file system**
- Still requires directory listing to plan jobs
  - **O(n) listing calls, n = # matching partitions**
  - **Eventual consistency breaks correctness**

🚫 表信息要去俩方找，性能差且一致性不好保证。

🚫 很多场景还不支持原子性

🚫 想迁S3省钱还搞不定

🚫 每个匹配的partition都要list，慢

Netflix: Hive的痛点

Less Obvious Problems

- Partition values are stored as strings
  - Requires character escaping
  - `null` stored as `__HIVE_DEFAULT_PARTITION__`

🚫 分区值要转义？太不优雅了

- HMS table statistics become stale
  - Statistics have to be regenerated manually

🚫 表的统计信息可能过期？

- A lot of undocumented layout variants

🚫 一堆搞不懂的变量？

- Bucket definition tied to Java and Hive

🚫 抽象没做好？

Netflix: Hive的痛点

Other Annoyances

- Users must know and use a table's physical layout
  - `ts > X` ⇒ full table scan!
  - Did you mean this?

```
ts > X and (d > day(X) or (d = day(X) and hr >= hour(X))
```

🚫 牢记物理分区 ？

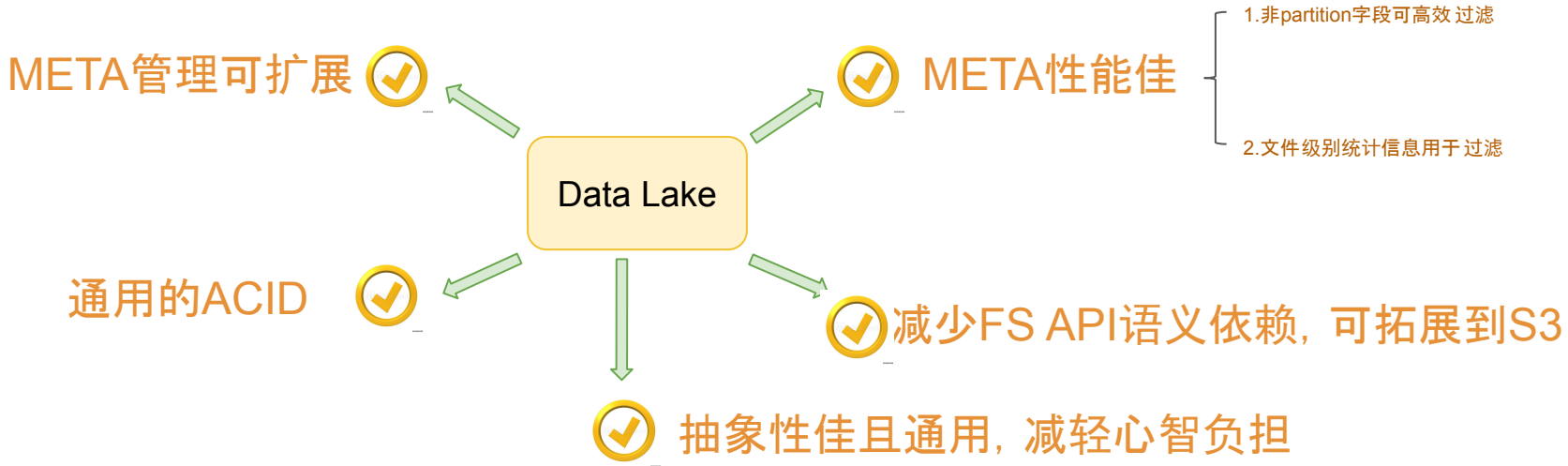
- Schema evolution rules are dependent on file format
  - CSV – by position; Avro & ORC – by name

🚫 DDL能不能搞，还要看文件格式？

- Unreliable: type support varies across formats
  - Which formats support decimal?
  - Does CSV support maps with struct keys?

🚫 数据类型支持，也要看文件格式？

Netflix 数据湖场景总结

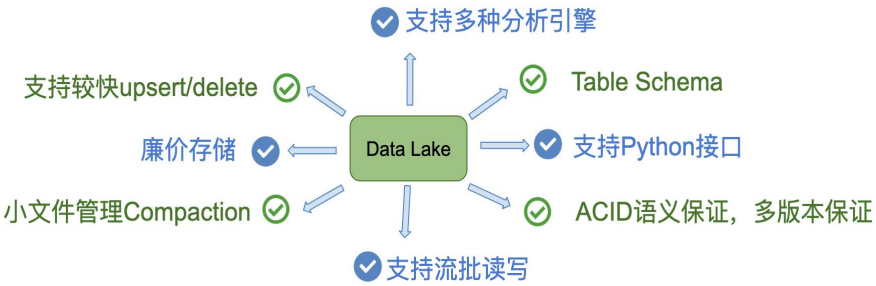


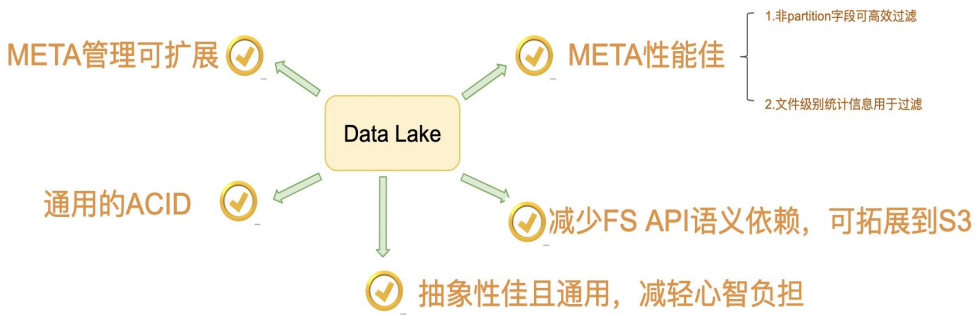
Netflix: Iceberg优化结果

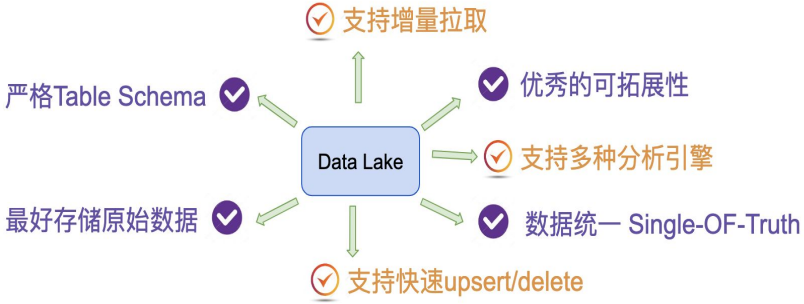
Atlas Historical Queries

- Hive table – with Parquet filters:
  - 400k+ splits, not combined
  - EXPLAIN query: **9.6 min** (planning wall time)
- Iceberg table – partition data filtering:
  - 15,218 splits, combined
  - **13 min** (wall time) / 61.5 hr (task time) / 10 sec (planning)
- Iceberg table – partition and min/max filtering:
  - 412 splits
  - **42 sec** (wall time) / 22 min (task time) / 25 sec (planning)

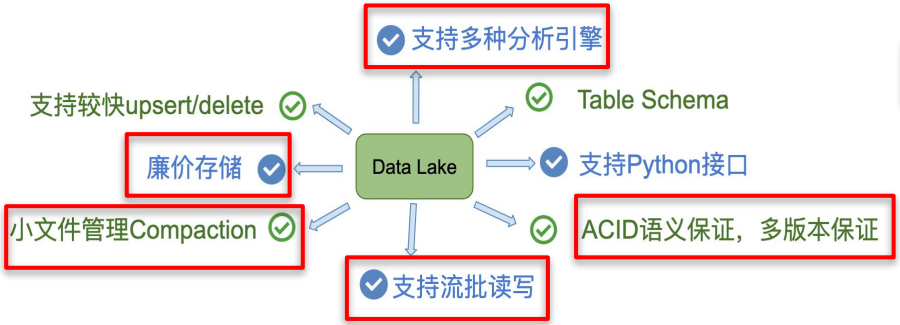
数据湖场景总结

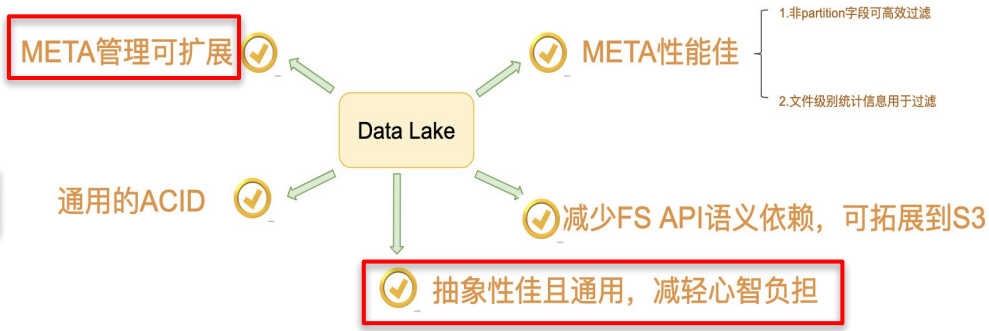

支持多种分析引擎
支持较快upsert/delete
Table Schema
廉价存储
Data Lake
支持Python接口
小文件管理Compaction
ACID语义保证，多版本保证
支持流批读写

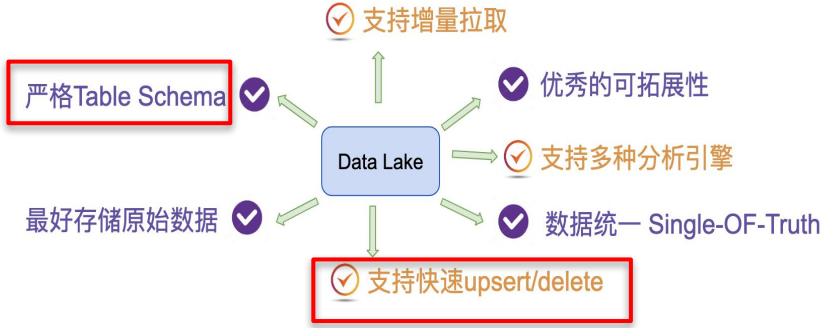

META管理可扩展
META性能佳
1.非partition字段可高效过滤
2.文件级别统计信息用于过滤
Data Lake
通用的ACID
减少FS API语义依赖，可拓展到S3
抽象性佳且通用，减轻心智负担


支持增量拉取
严格Table Schema
优秀的可拓展性
Data Lake
支持多种分析引擎
最好存储原始数据
数据统一 Single-OF-Truth
支持快速upsert/delete

数据湖场景总结


支持多种分析引擎
支持较快upsert/delete
Table Schema
廉价存储
Data Lake
支持Python接口
小文件管理Compaction
ACID语义保证，多版本保证
支持流批读写


META管理可扩展
META性能佳
1.非partition字段可高效过滤
2.文件级别统计信息用于过滤
Data Lake
通用的ACID
减少FS API语义依赖，可拓展到S3
抽象性佳且通用，减轻心智负担


支持增量拉取
严格Table Schema
优秀的可拓展性
Data Lake
支持多种分析引擎
最好存储原始数据
数据统一 Single-OF-Truth
支持快速upsert/delete

- 数据湖用户场景分析
  - Databricks: Lambda架构问题及改进 (link)
  - Uber: 城市运营人员要求更实时的行程数据 (link)
  - Netflix: Hive的痛点 (link)
- **数据湖产品对比**
  - **8个维度深度对比Delta、Hudi、Iceberg**
  - 选型 ？
- 其他
  - Kafka + KSQL
  - AWS、Azure、Google云数据湖现状

Solution	Updates Deletes	Compaction Cleanup	File Formats	Engines	SQL DML	Write Amplification	Open Source Governance
Delta.io (Databricks)	Yes	• Manual Cleanup • No Compaction	Parquet	• Spark • Presto	No	High	No
Apache Iceberg	No	None	• Parquet • ORC	• Spark • Presto	No	Unknown	Yes
Apache Hudi	Yes	Automatic	• Parquet • Avro	• Spark • Presto • Hive	No	Low	Yes
Apache Hive ACID	Yes	Automatic and Manual	ORC	• Hive • Spark (LLAP)	Yes	Low	Yes

Contrast#1 ACID & Isolation level

Solution	ACID Support	Isolation Level	Concurrent Multi-Writers	Time Travel
Iceberg	Yes	Write Serialization	No	Yes
Hudi	Yes	Snapshot Isolation	Yes	Yes
Open Delta	Yes	Serialization Write Serialization Snapshot Isolation	Yes	Yes
Hive ACID	Yes	Snapshot Isolation	Yes	No

Contrast#2 Schema Enforcement & Evolution

Solution	Schema Evolution	Self-defined schema object
Iceberg	all	Yes
Hudi	back-compatible	No(spark-schema)
Open Delta	all	No(spark-schema)
Hive ACID	all	No(Hive-schema)

Contrast#3 Unify Batch & Streaming

Solution	Batch Read	Batch Write	Streaming Read	Streaming Write
Iceberg	Yes (pig/spark) Hive?	Yes (spark)	Ongoing(issue#179)	Yes
Hudi	Yes (RO-view: hive/spark/presto)	Yes (spark)	Yes	Yes
Open Delta	Yes (hive/spark/presto)	Yes (spark)	Yes	Yes
Hive ACID	Yes	Yes	No	Yes(few user-cases)

Contrast#4 Abstractions & Pluggable

Solution	Engine Pluggable (Write Path)	Engine Pluggable (Read Path)	Storage Pluggable (Less Storage API Binding)	Open File Format
Iceberg	Yes	Yes	Yes	Yes
Hudi	No (Bind with spark)	Yes	Yes	Yes(Data) + No(Log)
Open Delta	No (Bind with spark)	Yes	Yes	Yes
Hive ACID	Yes	Yes	No	No(Only ORC)

Contrast#5 Performance & Query Optimization

Solution	Filter PushDown	Low meta cost	Indexing within partitions (Boost the perf of selective queries)	CopyOnWrite	MergeOnRead	Auto-Compaction
Iceberg	Yes	Yes	Road-map	Yes	On-going	No
Hudi	No	Yes	-	Yes	Yes	Yes
Open Delta	No	Yes	-	Yes	No	No
Close Delta	Yes	Yes	Yes	Yes	Yes	Yes
Hive ACID	Yes	No	-	No	Yes	Yes

Contrast#6 usability & others

Solution	One line demo	Python Support	File Encryption	Cli Command
Iceberg	Not Good	Yes	Yes	No
Hudi	Medium	No	No	Yes
Open Delta	Good	Yes	No	Yes
Close Delta	Good	Yes	No	Yes
Hive ACID	Medium	No	No	Yes

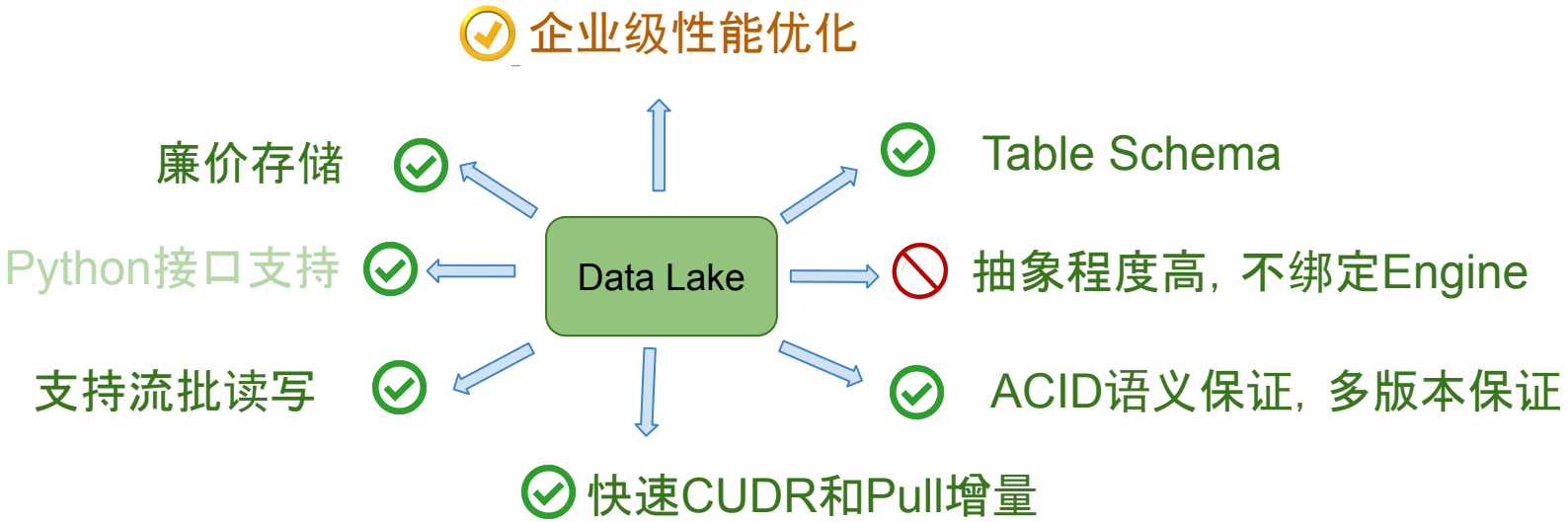
Contrast#7 Community Status (until 2020-01-08)

Iteams	Delta	Iceberg	Hudi
Open Source Time	2019/04/12	2018/11/06(incubation)	2019/01/17(incubation)
Github Watch	136	43	1.3K
Github Star	2K	310	920
Github Fork	404	139	355
Github Issue	65	100	11
Apache JIRA	-	-	484
Github Open PR	18	25	38
Commits	250	644	714
Contributors	42	60	75
Committers	-	8	13
PMCs	-	8	10
Releases	5	1	37

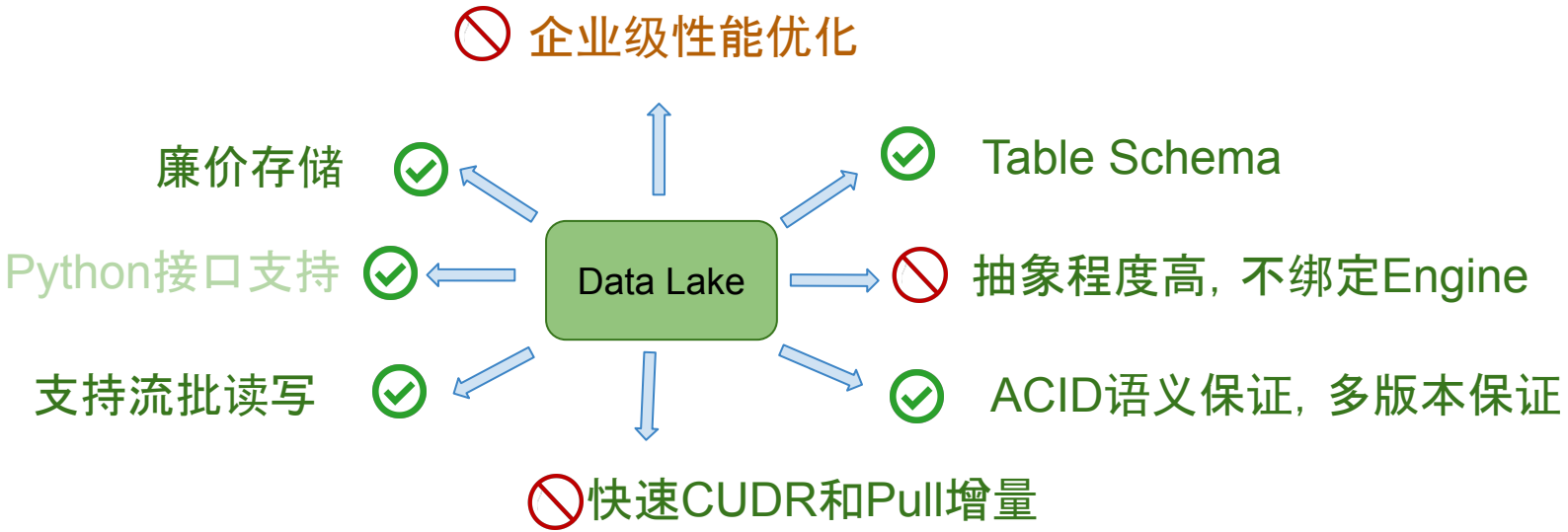
Contrast#8 Selection of other teams

Solution	AWS EMR	Uber	Databricks	Netflix	Apple	Tencent	Apache Flink
Iceberg				✓	✓	✓	
Hudi	✓	✓					
Delta	✓		✓				
Hive ACID							

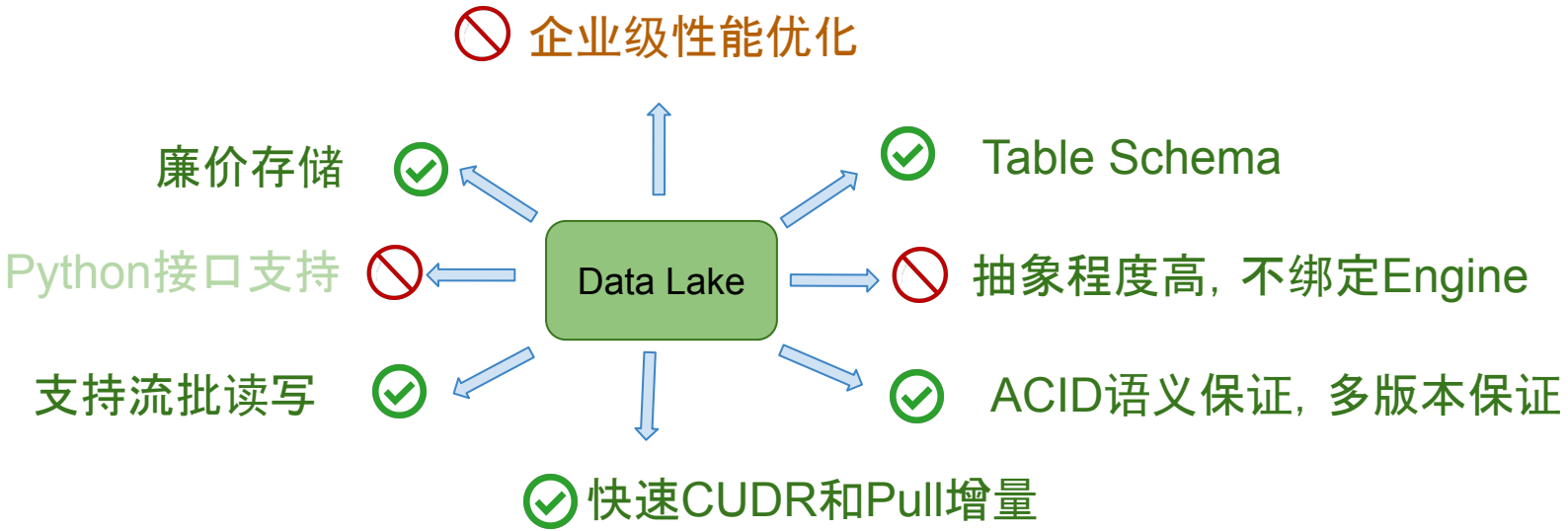
Databricks Delta



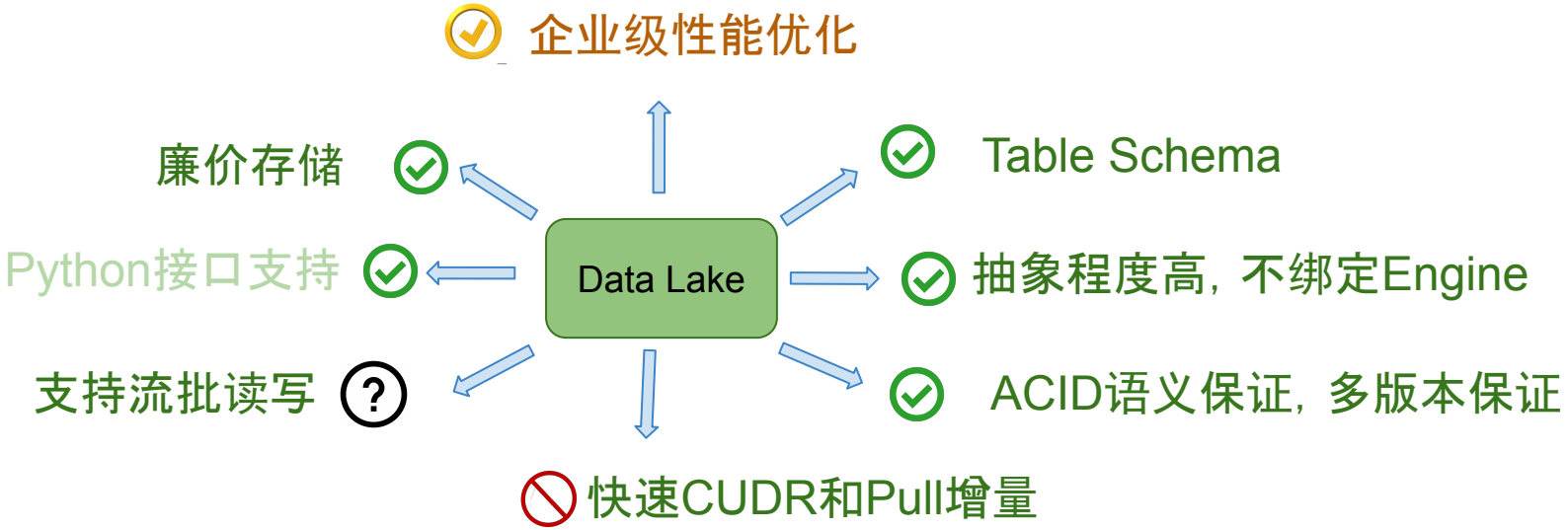
Open Source Delta



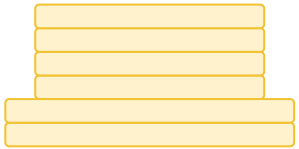
Apache Hudi


企业级性能优化
廉价存储
Table Schema
Python接口支持
Data Lake
抽象程度高，不绑定Engine
支持流批读写
ACID语义保证，多版本保证
快速CUDR和Pull增量

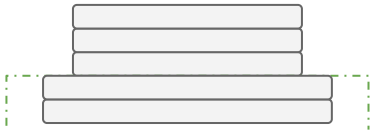
Apache Iceberg



Summary



Delta



Apache Iceberg

Apache Hudi

Apache Hive ACID

- 数据湖用户场景分析
  - Databricks: Lambda架构问题及改进 (link)
  - Uber: 城市运营人员要求更实时的行程数据 (link)
  - Netflix: Hive的痛点 (link)
- 数据湖产品对比
  - 8个维度深度对比Delta、Hudi、Iceberg
  - 选型是 ？
- **其他**
  - Kafka + KSQL
  - AWS数据湖现状

Kafka+KSQL

Message Queue | Streaming Compute

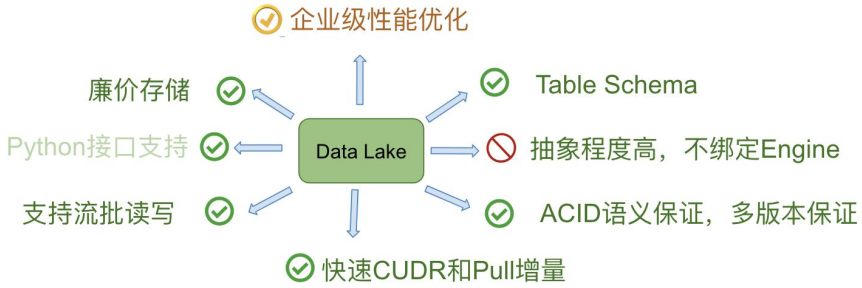
Message Queue | Streaming Compute | Batch, AI etc

AWS数据湖产品现状

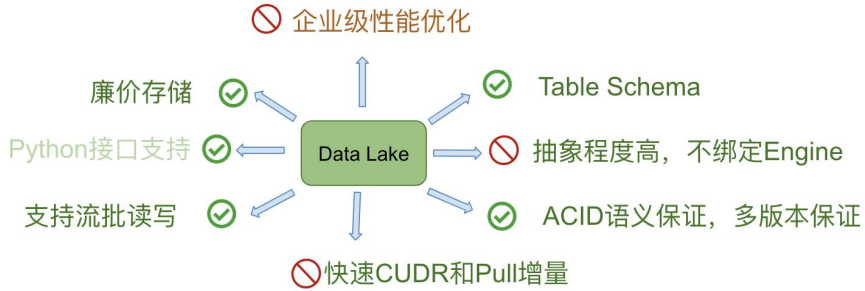
- Data Lake On AWS (#1 #2)
  - AWS Glue
  - AWS Lake Information
- AWS EMR
  - Delta/Hudi

Thank You

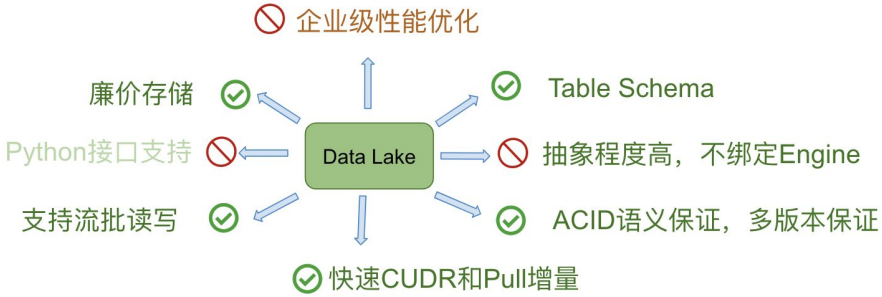
Databricks Delta



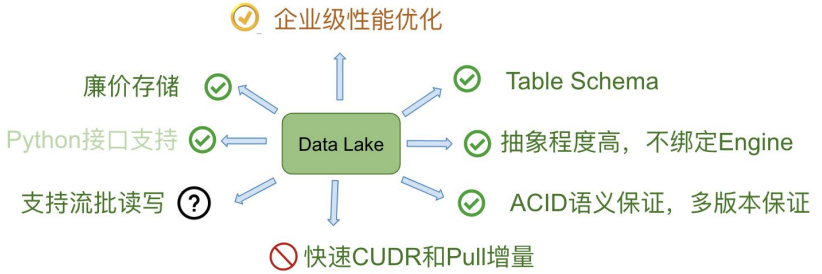
Open Source Delta



Apache Hudi



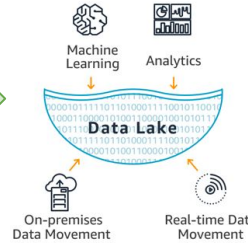
Apache Iceberg



AI training

Events

Real-time computing/Transforming



Real-time Analytics

Batch / Interactive Computing

Business Reporting