THE UNIVERSITY *of* EDINBURGH
informatics

Serverless-Native Analytics

UK Systems 2023

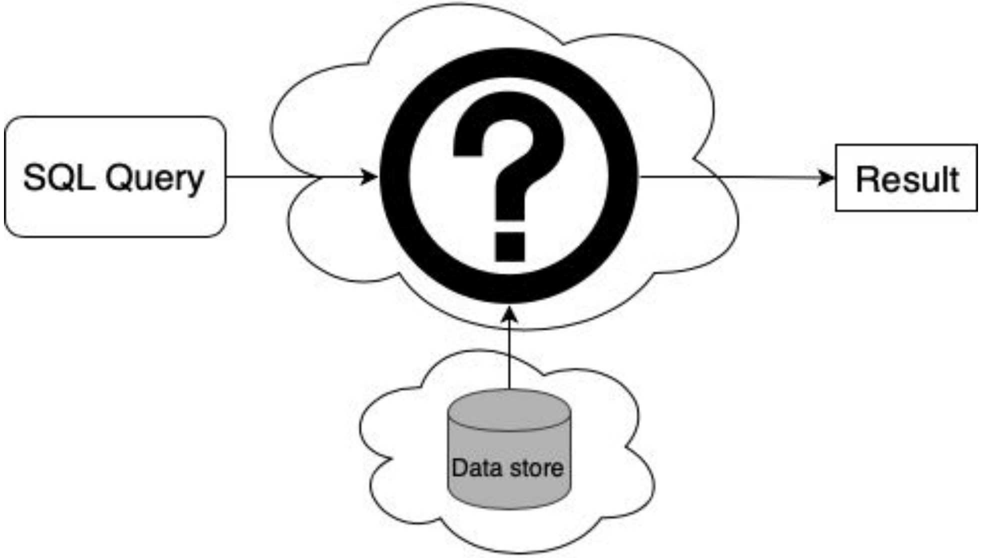
Co-authors: Shengda Zhu & Shyam Jesalpura

Supervised by: Boris Grot, Antonio Barbalace, Amir Shaikha

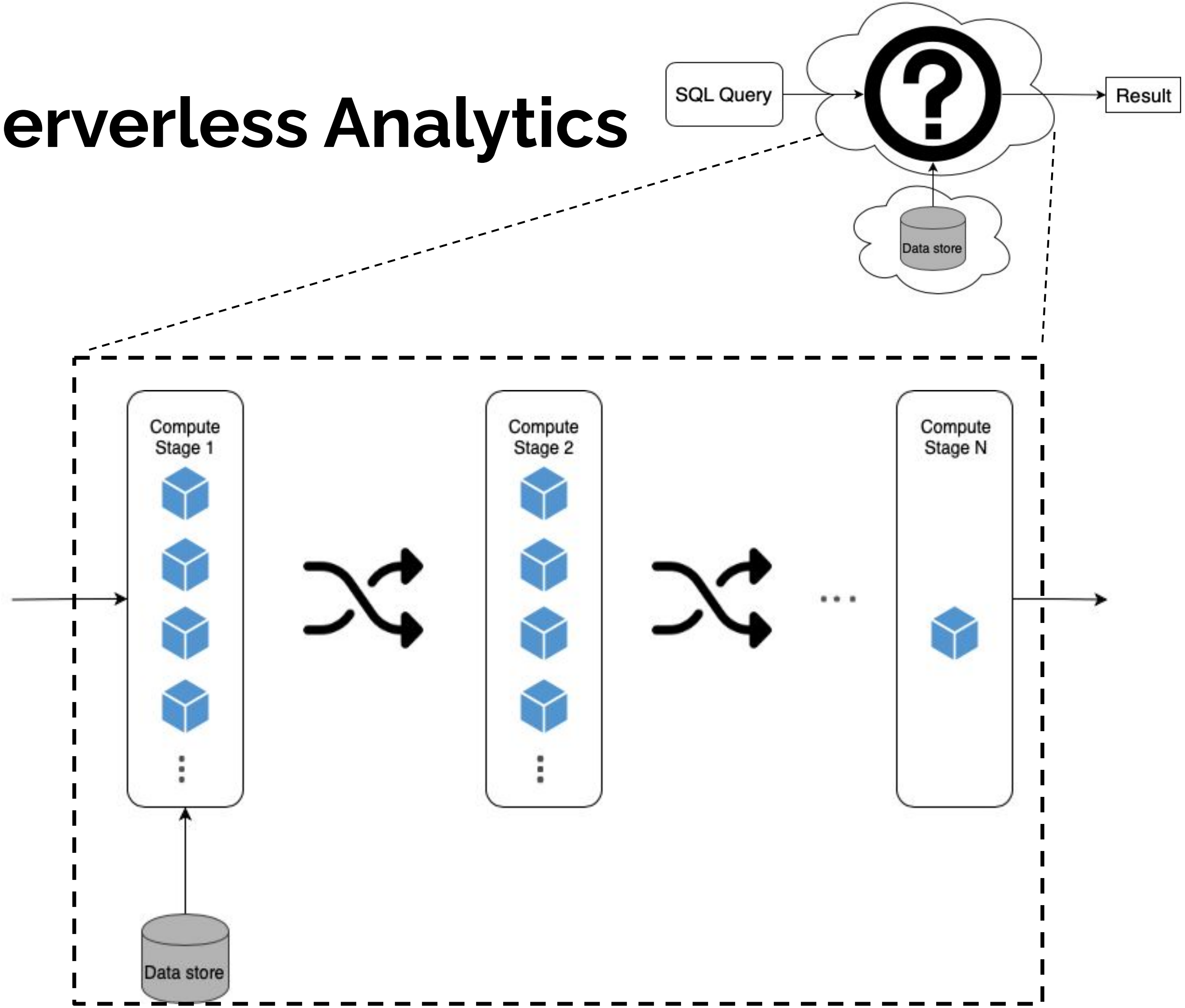
Toward Serverless Analytics

Analytics: computational analysis of data

- SQL queries are widely used → e.g., select count(*) from Cars c where c.color=‘blue’
- Resource-intensive: compute + memory + storage

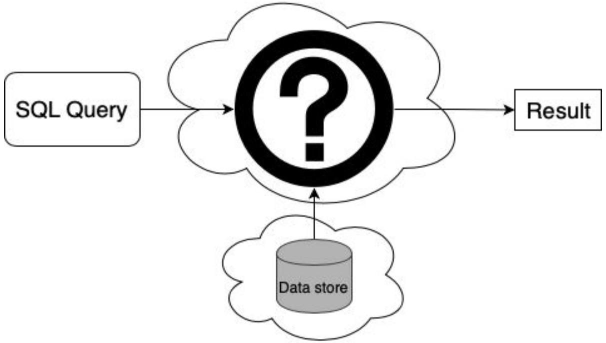


Toward Serverless Analytics



🟦 Computing Node

Toward Serverless Analytics



Today’s computing node types:

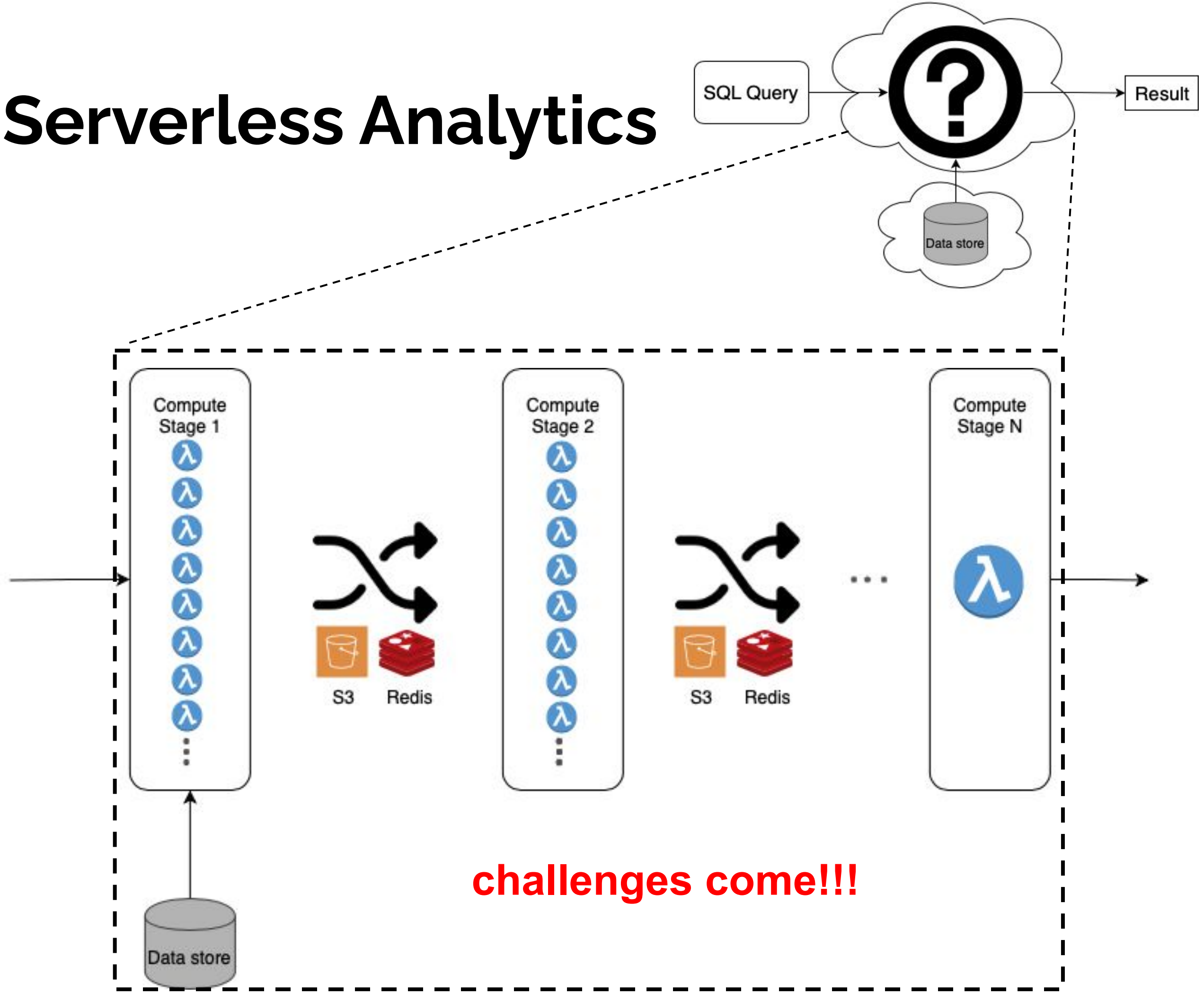
VM-based (e.g., Amazon EC2):

- Slow to scale 🐢
- Always-on → 💰💰💰

Serverless (e.g., AWS Lambda): λ

- On-demand
- Extremely elastic but …
- Limited execution time
- Stateless
- No direct communication

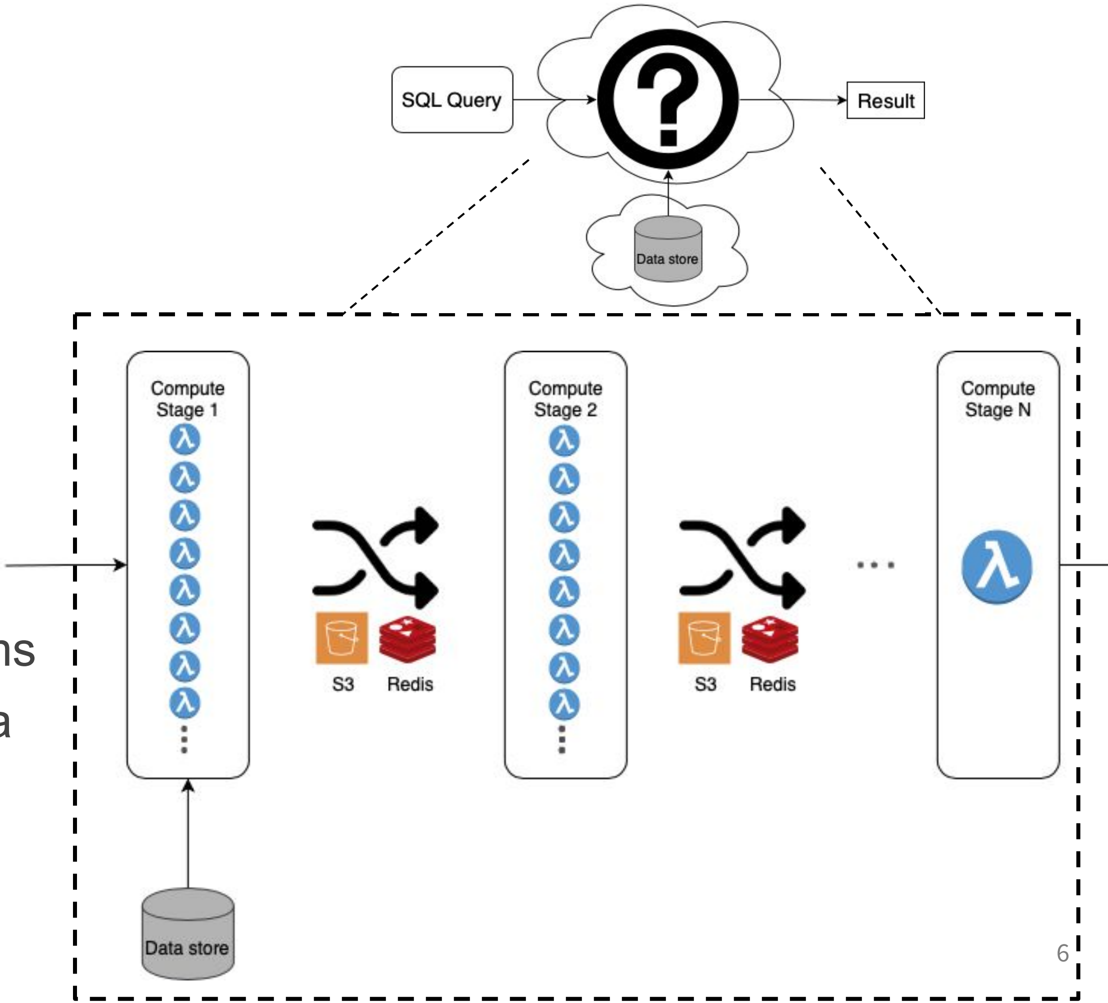
Toward Serverless Analytics



challenges come!!!

Existing Solutions:

- Focus on partial problems
  - Data transfer
  - Storage
  - Computing
  - …
- Hard-coded for certain queries
- Entirely closed-source platforms
  - e.g. Amazon EMR, Athena

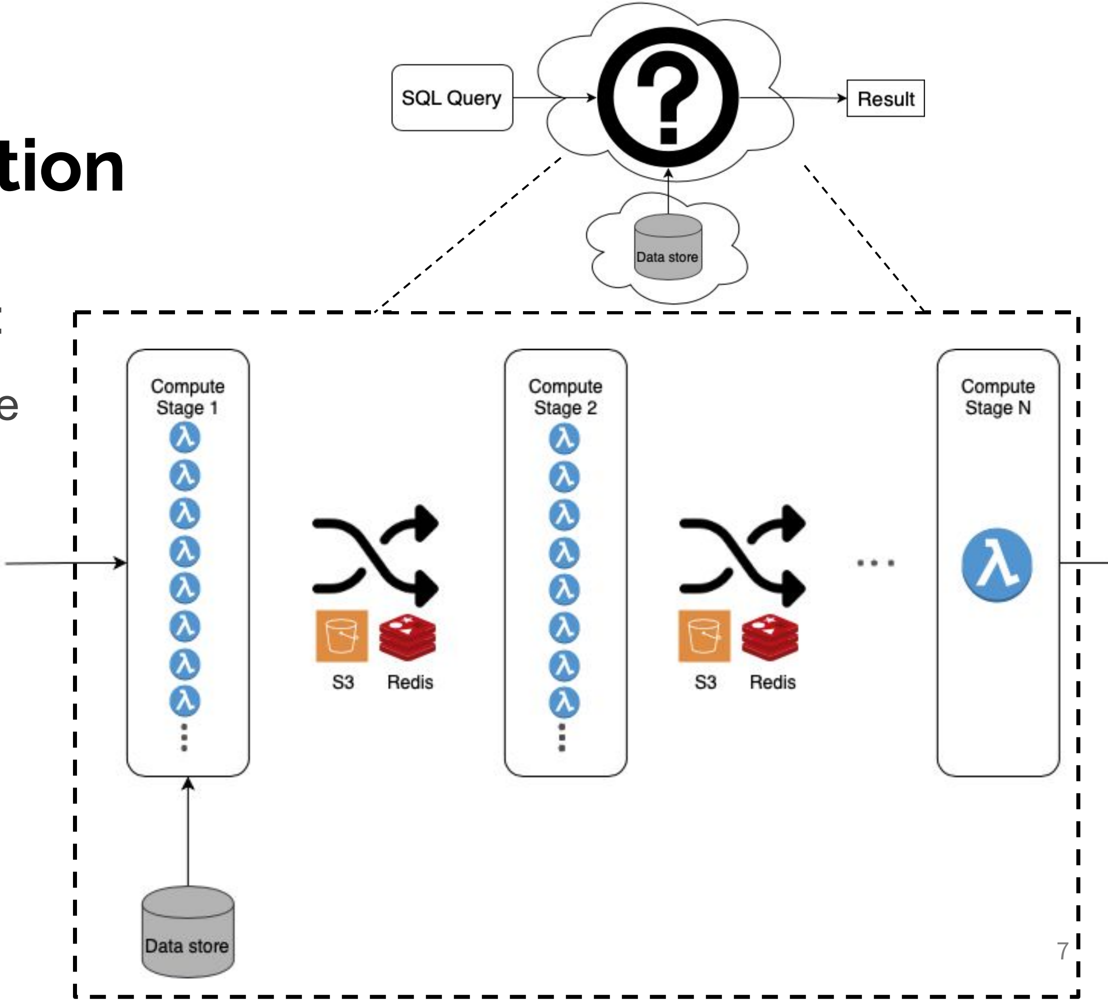


Challenge:
End-to-end Optimization

Large & complex optimization space:

- Number of workers and worker type at each stage
- Intermediate storage type (e.g., S3 vs in-memory cache)
- Communication options
- …

Not independent but interrelated!

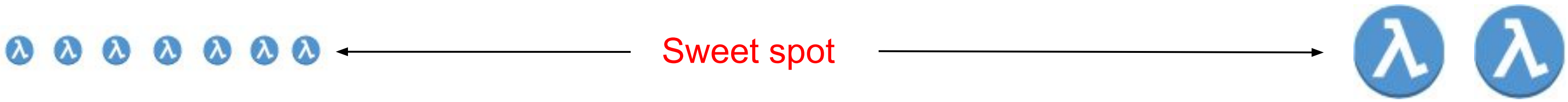


Motivational Example

Query time = CPU time + communication time

Trade-offs: worker type, #worker and communication overhead

- Using "fat" workers (more cores, more memory) → less workers → less data to transfer. But…
- Larger chunks of data per worker → Long I/O latency
- Fat workers lead to overprovision easily → 💰 💰 💰

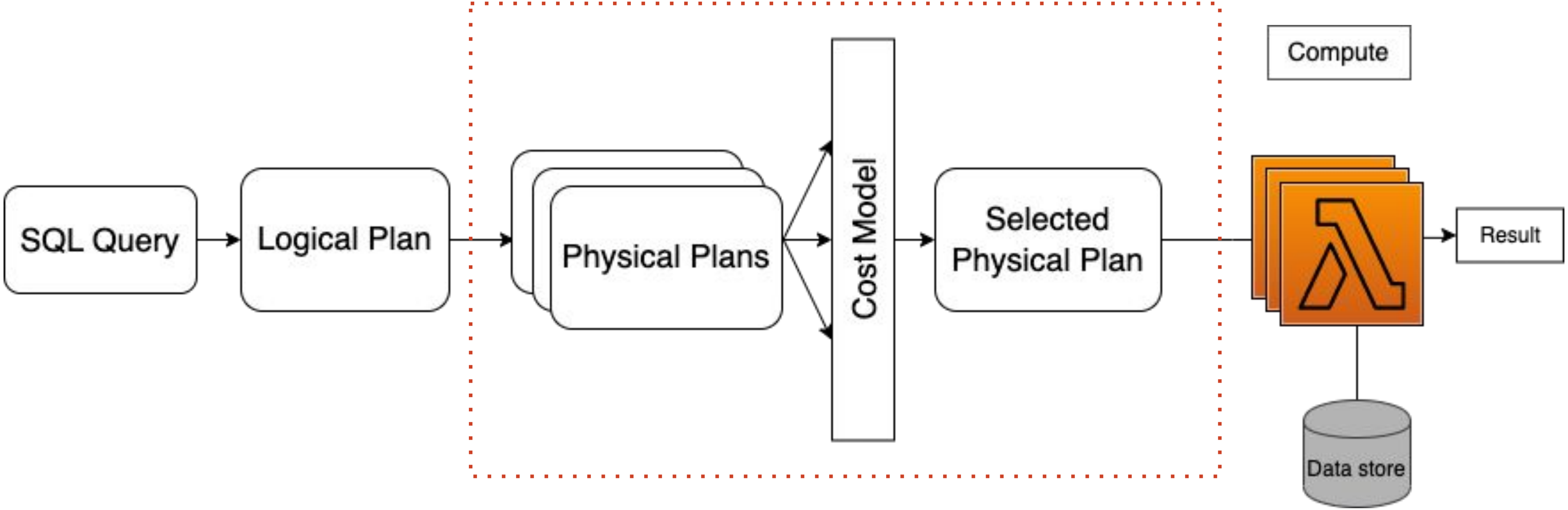


Sweet spot

Need an open, serverless-native analytics platform

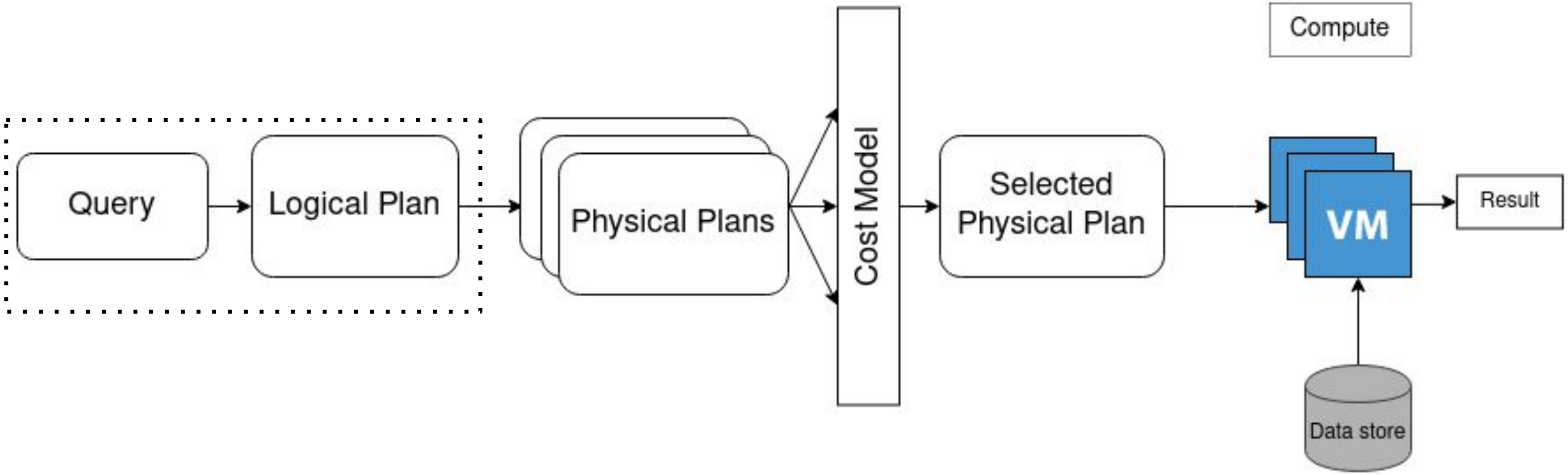
Insight: Serverless Analytics are Still Analytics

Can apply established query-optimization techniques!

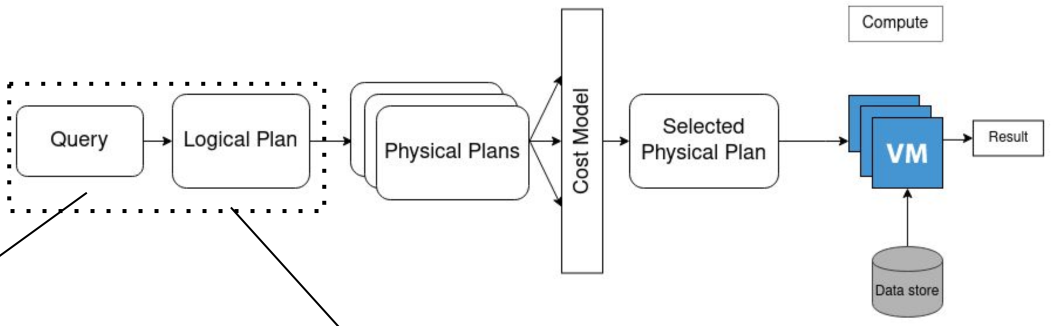


Must specialize to the serverless context!

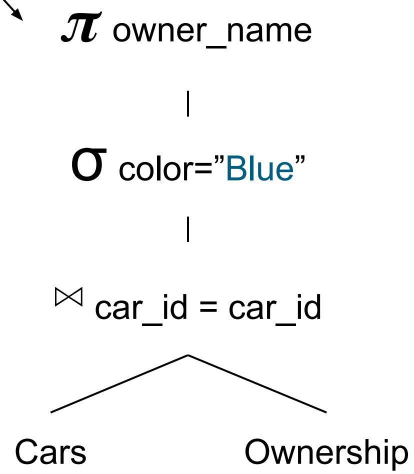
Life of a Query



Life of a Query



```
SELECT owner_name
FROM Cars c, Ownership o
WHERE c.car_id = o.car_id
and c.color = 'blue'
```



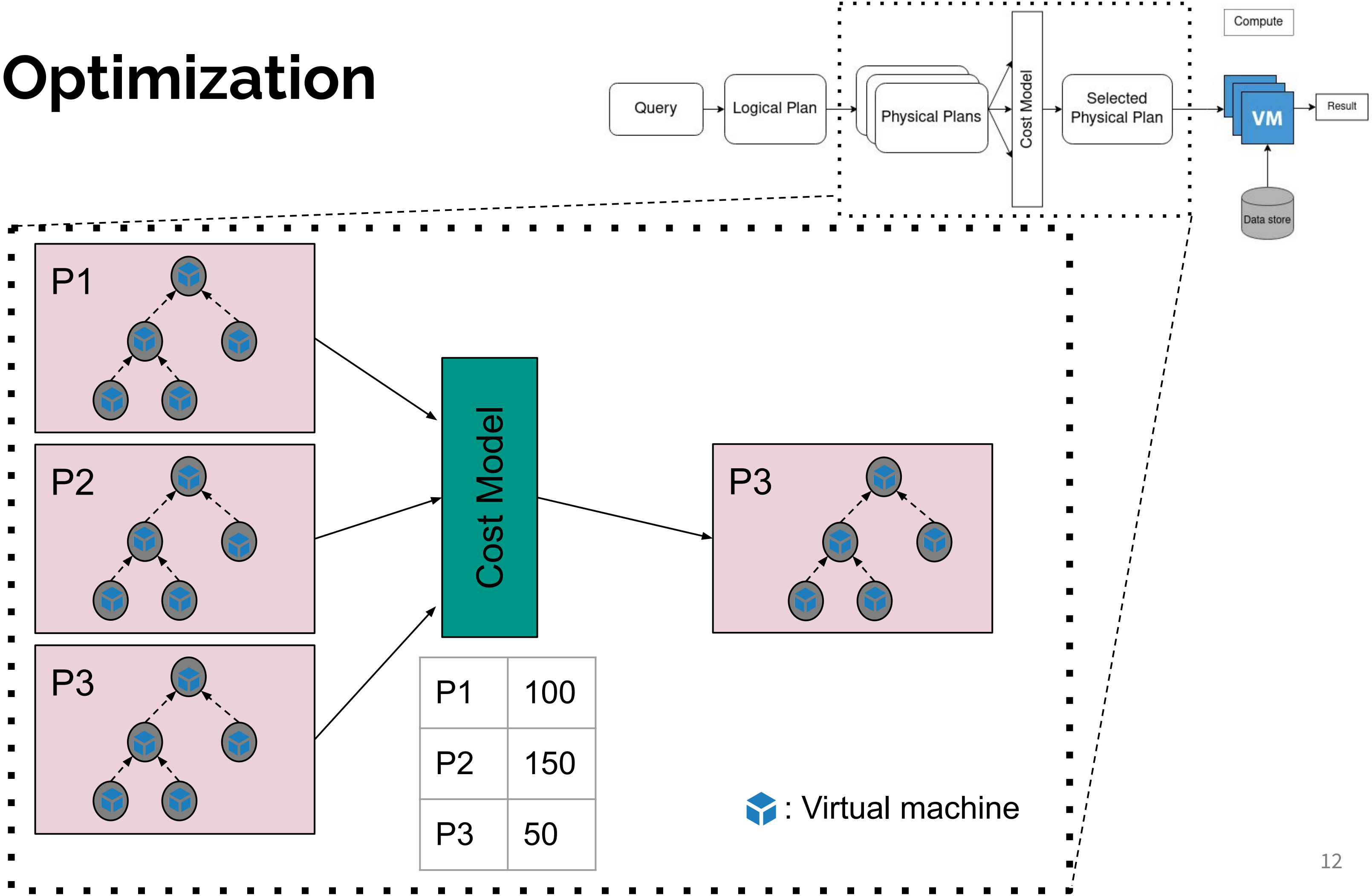
π owner_name

σ color="Blue"

$\bowtie$ car_id = car_id

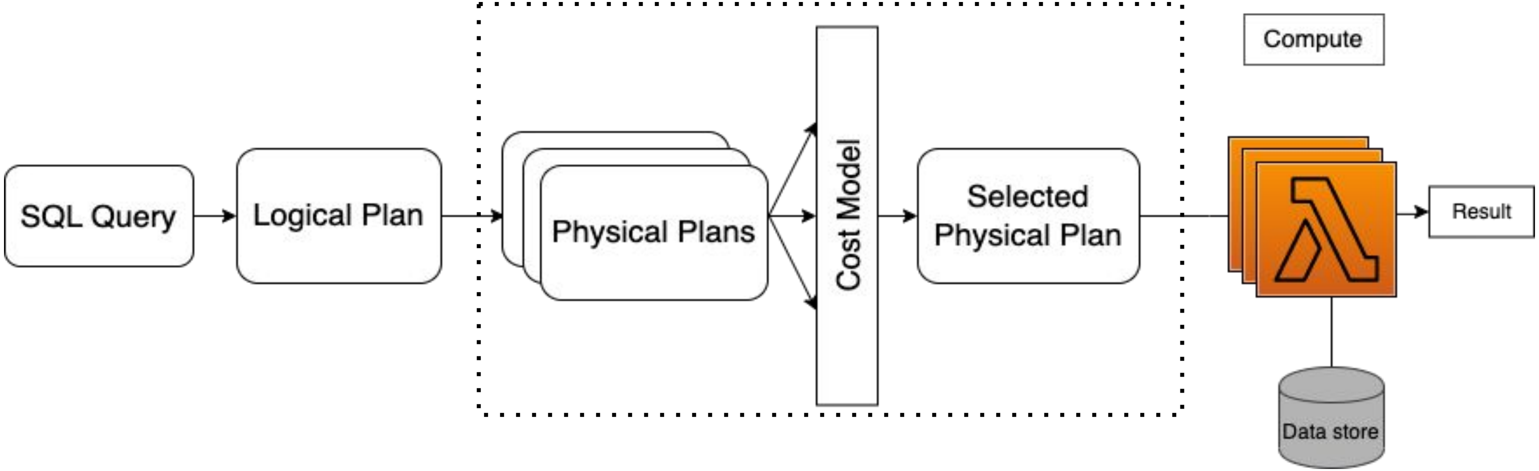
Cars Ownership

Query Optimization

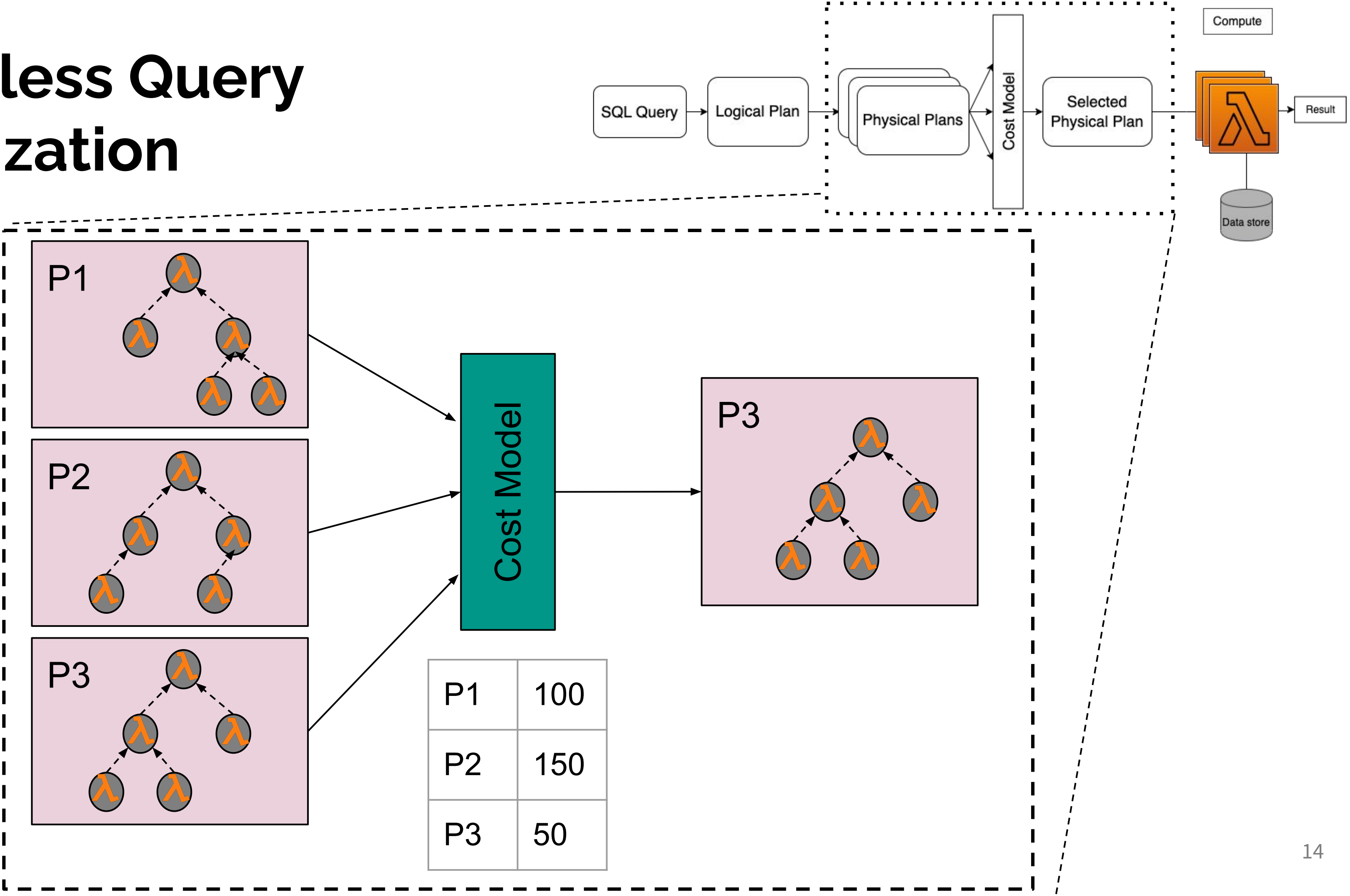


P1	100
P2	150
P3	50

Serverless Query Optimization

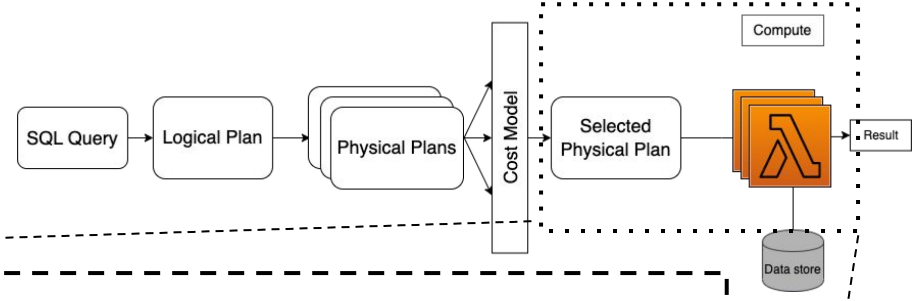


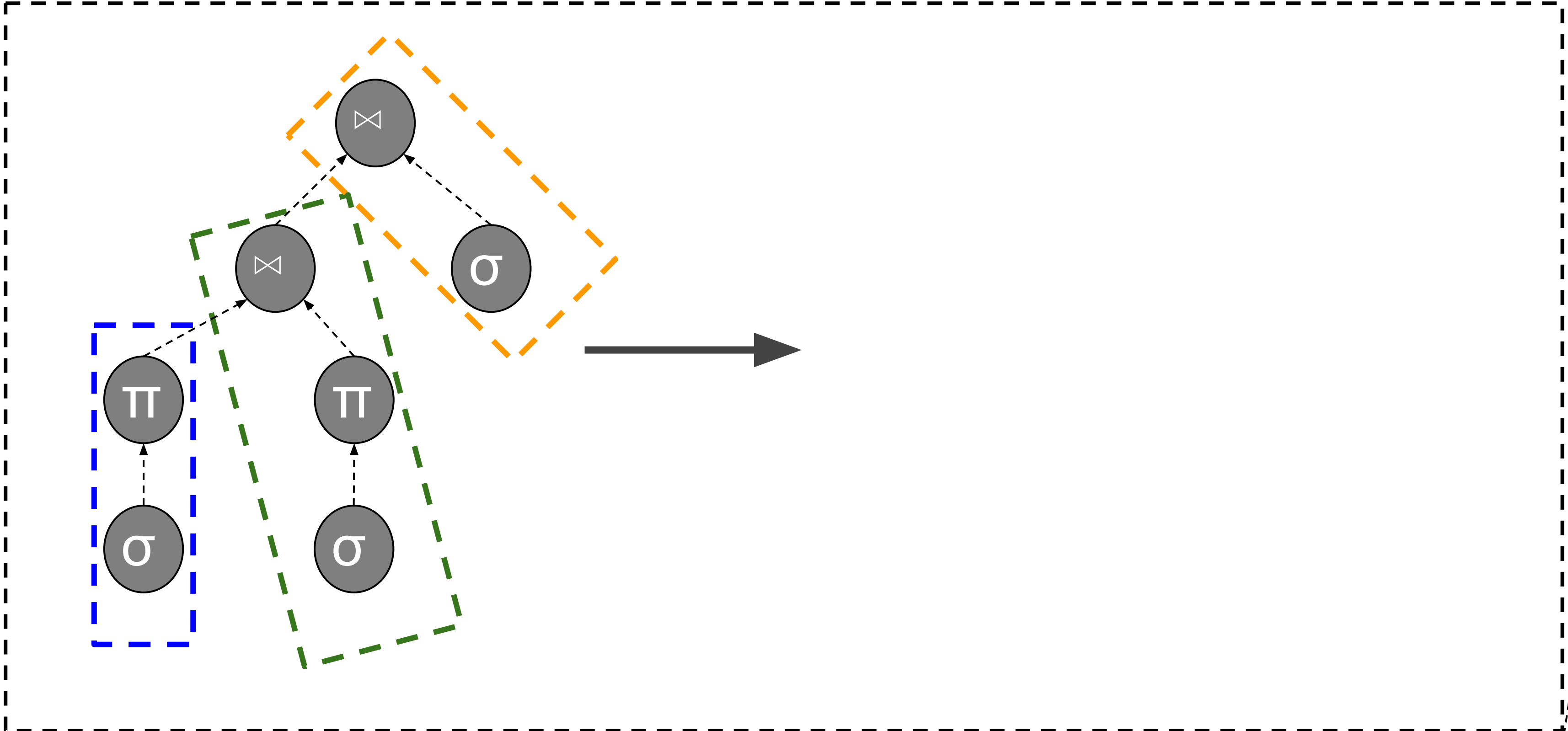
Serverless Query Optimization



P1	100
P2	150
P3	50

Query Execution





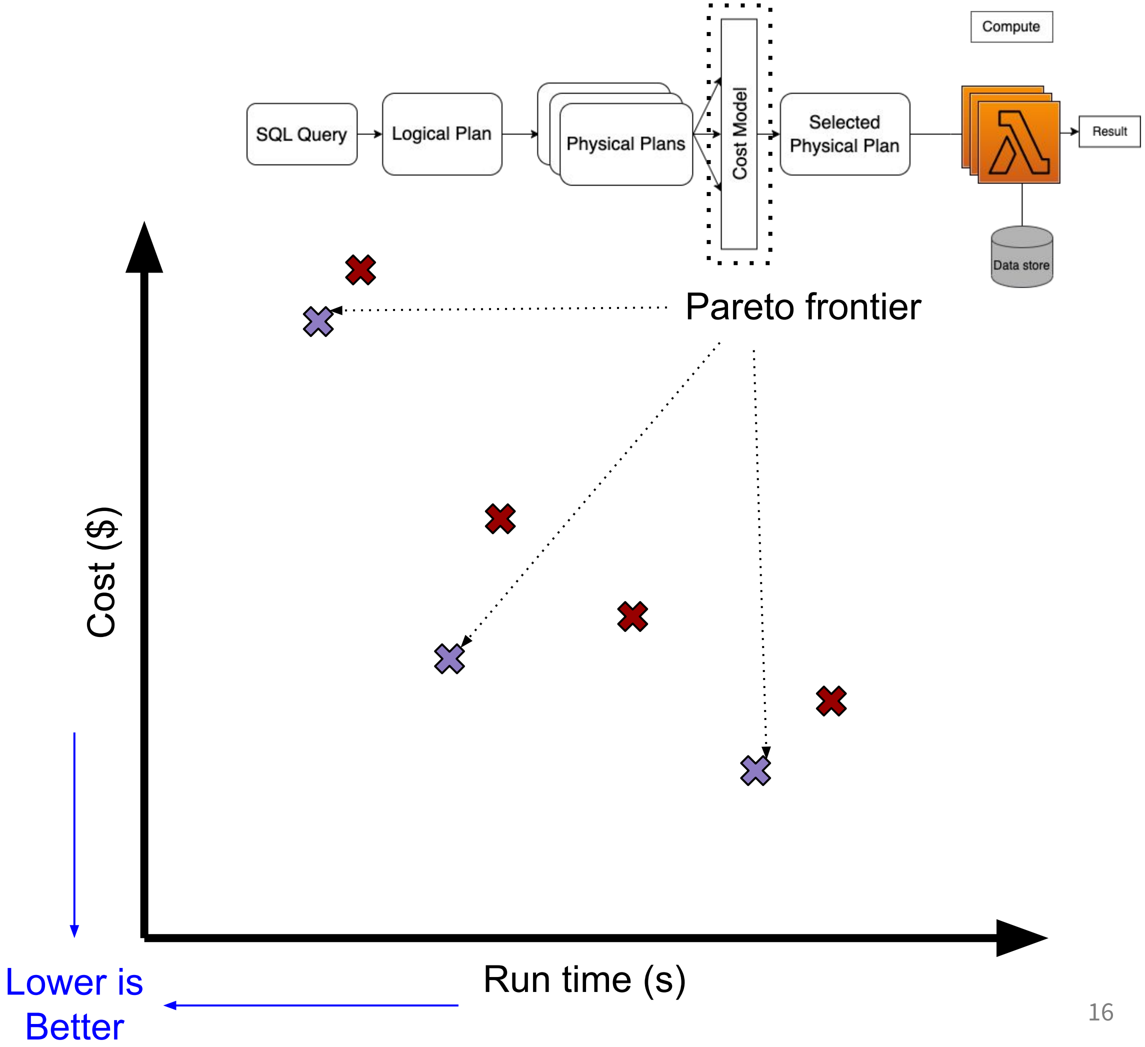
Optimisation Space

Generic:

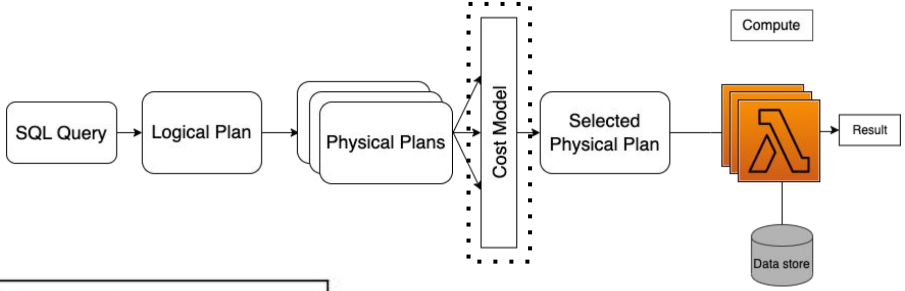
- Storage
- Communication
- Computation

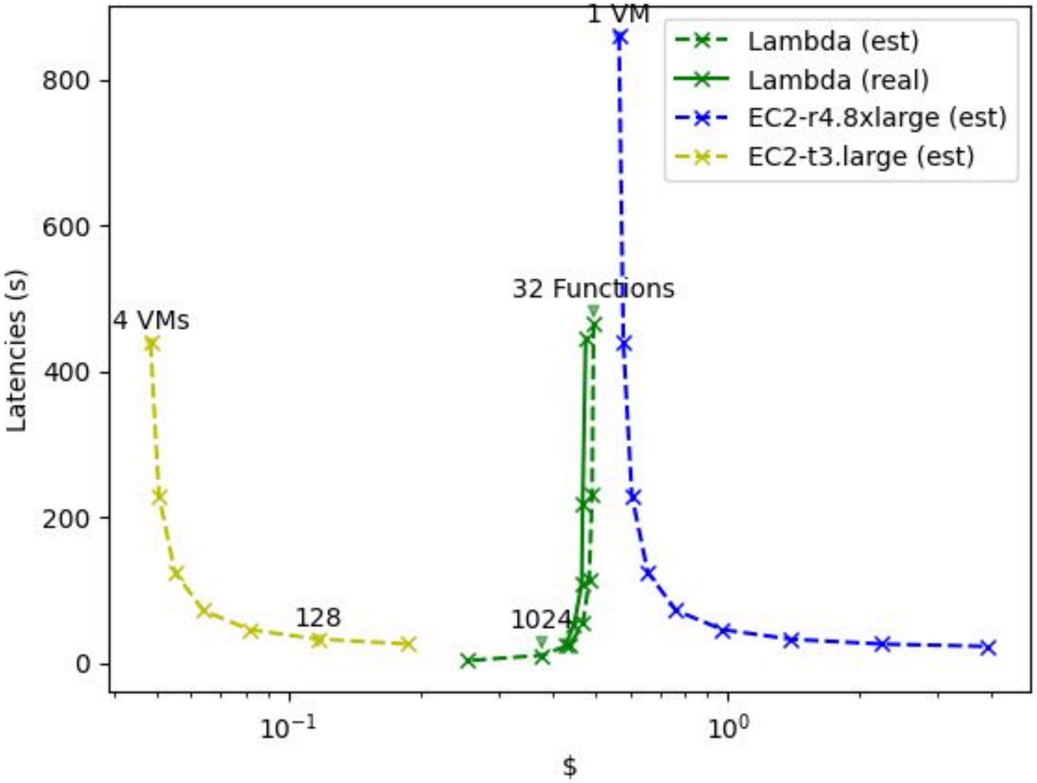
Query-specific:

- Cardinality
- Selectivity
- Parallelism

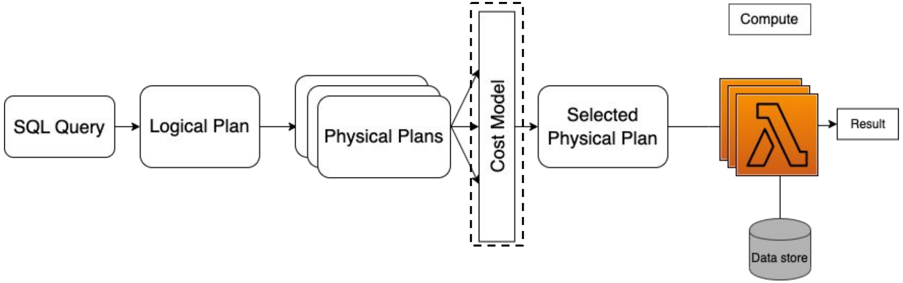


WordCount on 1TB data





Progress



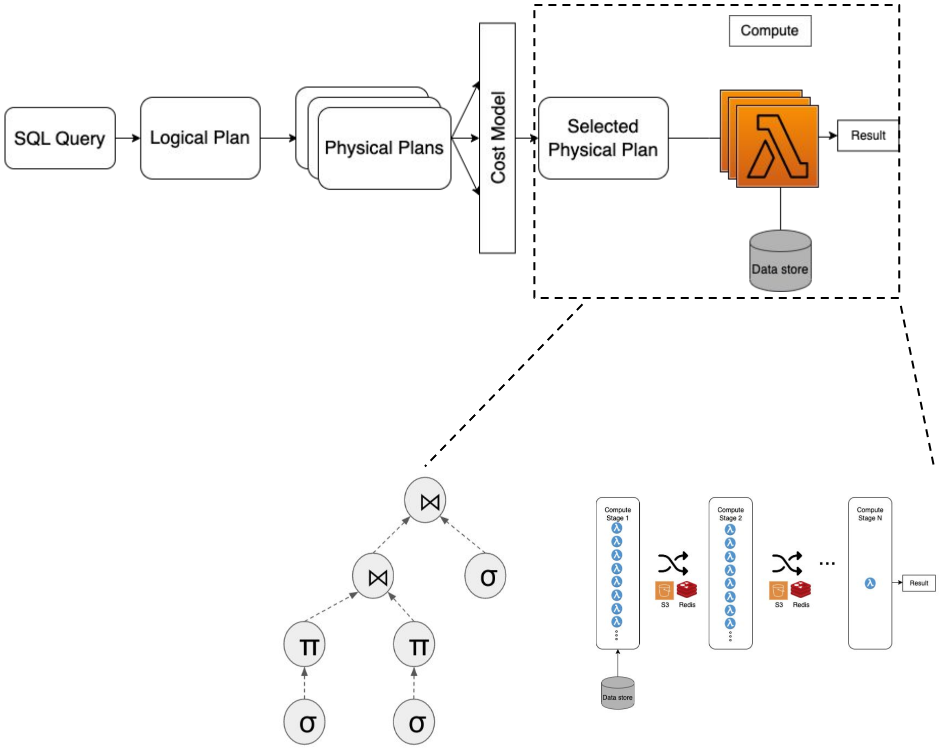
Cost Model

- Benchmarking storage services (S3/Elasticache)
- Benchmarking compute: Serverless/VMs
    - Network BW
    - Startup time

Progress

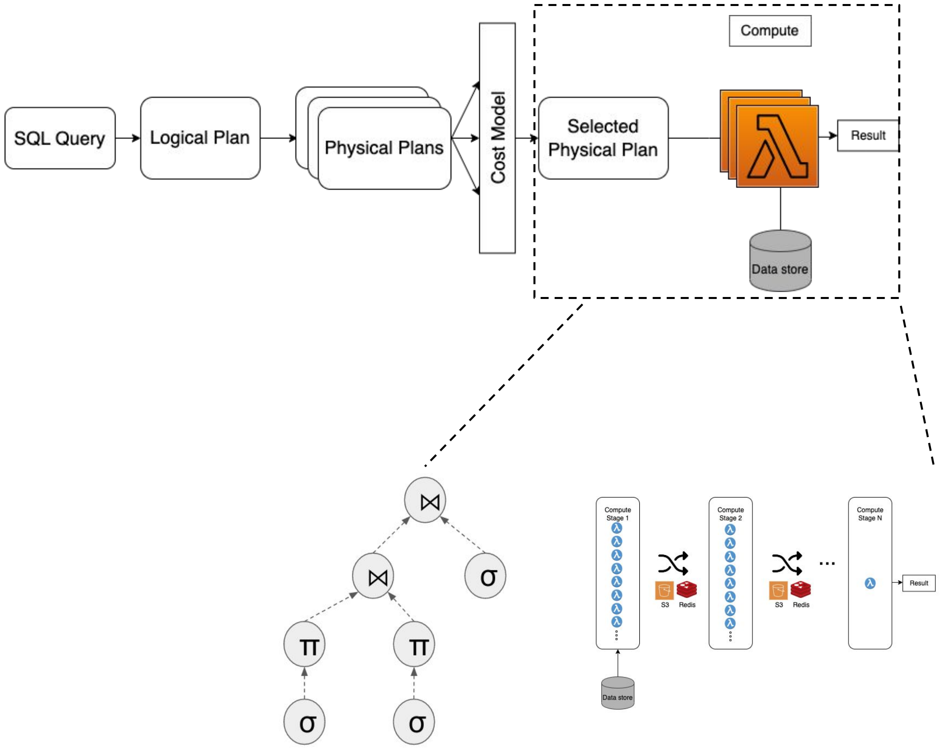
Operators:

- Scan, filter, aggregate



Next steps

Add exchange operator for shuffle/map-reduce/joins



Summary

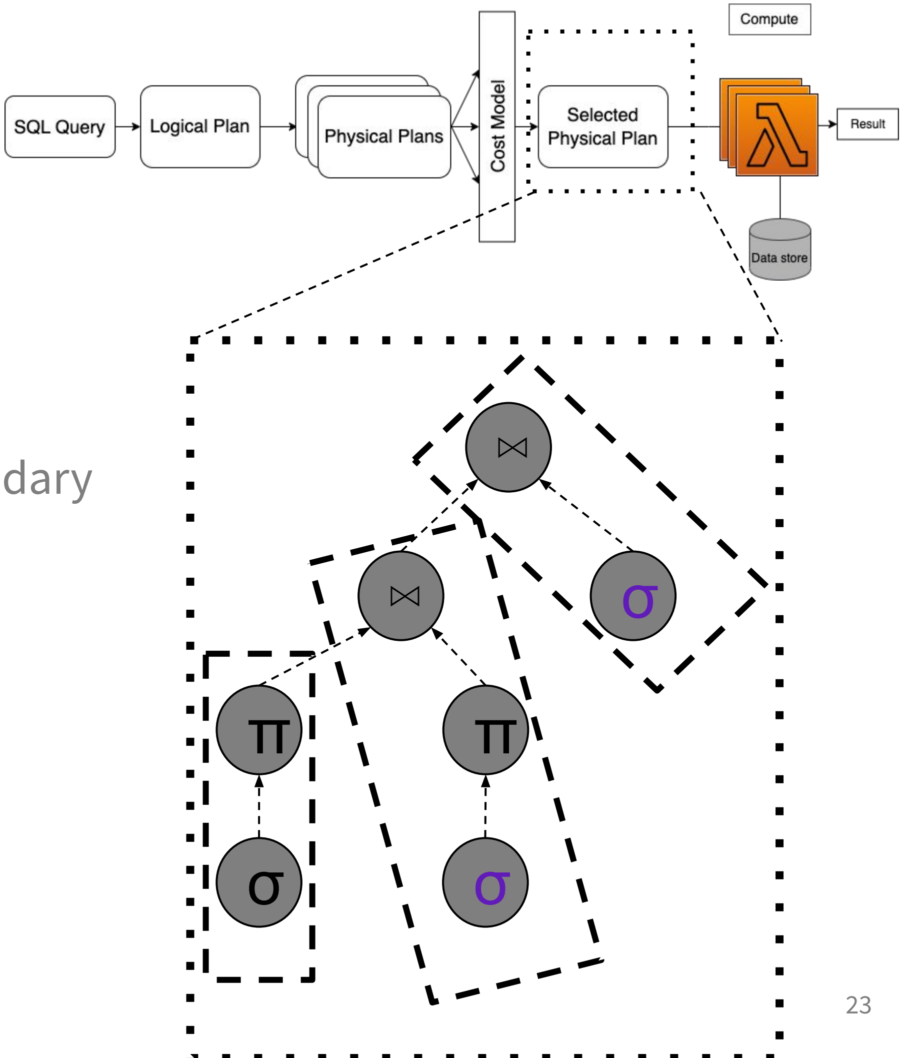
Serverless workers + query planning → serverless native analytics 🥳

Thank You for your attention

Contacts:
shyam.jesalpura@ed.ac.uk
shengda.zhu@ed.ac.uk

Query Plan

- Pipeline
  - Stream data without communication
  - Selection, projection
  - Mapped to one lambda
- Pipeline-breakers → communication boundary
  - Aggregations, joins
- Scan operator
  - S3, Parquet
- Communication
  - Storage, Redis, etc.
- Worker capabilities
  - Count, # of cores, memory
- Producer-consumer model (push engine)

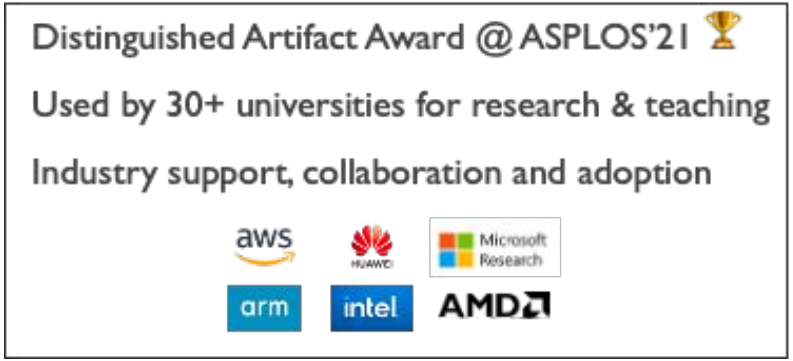


vHive: our open-source serverless stack

Representative of today's clouds

- ○ Industry-grade technologies
- ○ Knative FaaS API, Firecracker & gVisor MicroVMs, Kubernetes
- ○ First to support Firecracker snapshots at scale

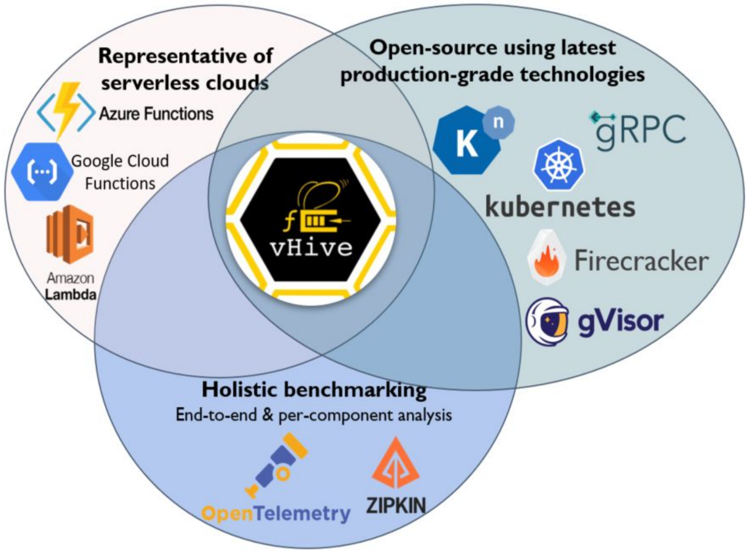
State-of-the-art performance analysis tools


Distinguished Artifact Award @ ASPLOS'21 🏆

Used by 30+ universities for research & teaching

Industry support, collaboration and adoption

aws
HUAWEI
Microsoft Research
arm
intel
AMD



github.com/vhive-serverless/vhive

● Go ☆ 210 61