

Hello!

CS 744: F1

Shivaram Venkataraman

Spring 2025

ADMINISTRIVIA

- Course Project Proposal: Due soon! → Due tomorrow
↳ Change it on Canvas
- Midterm details are on Piazza.

↳ Next Tuesday

OH at 3pm

Applications

Machine Learning

SQL

Streaming

Graph

Computational Engines

Scalable Storage Systems

Resource Management

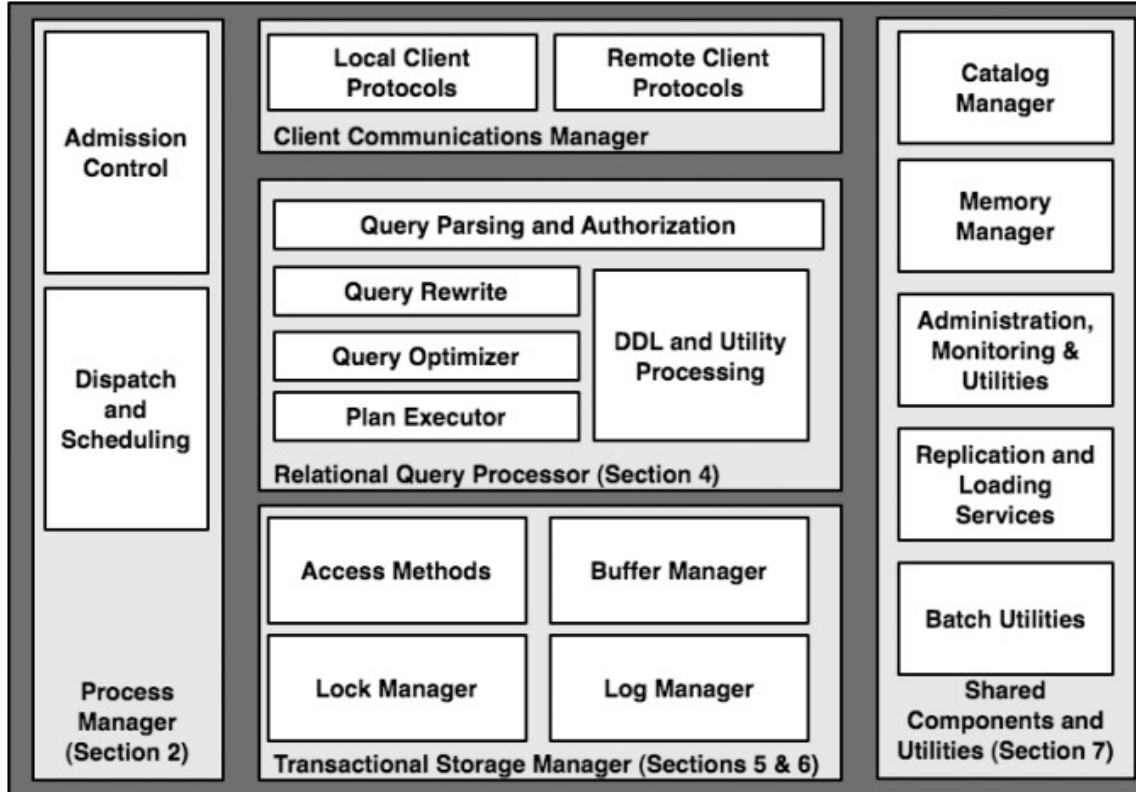


Datacenter Architecture



SQL: STRUCTURED QUERY LANGUAGE

DATABASE SYSTEMS



Scale
↳ large amounts of data

F1

Deployment

↳

datacenter

cloud-based

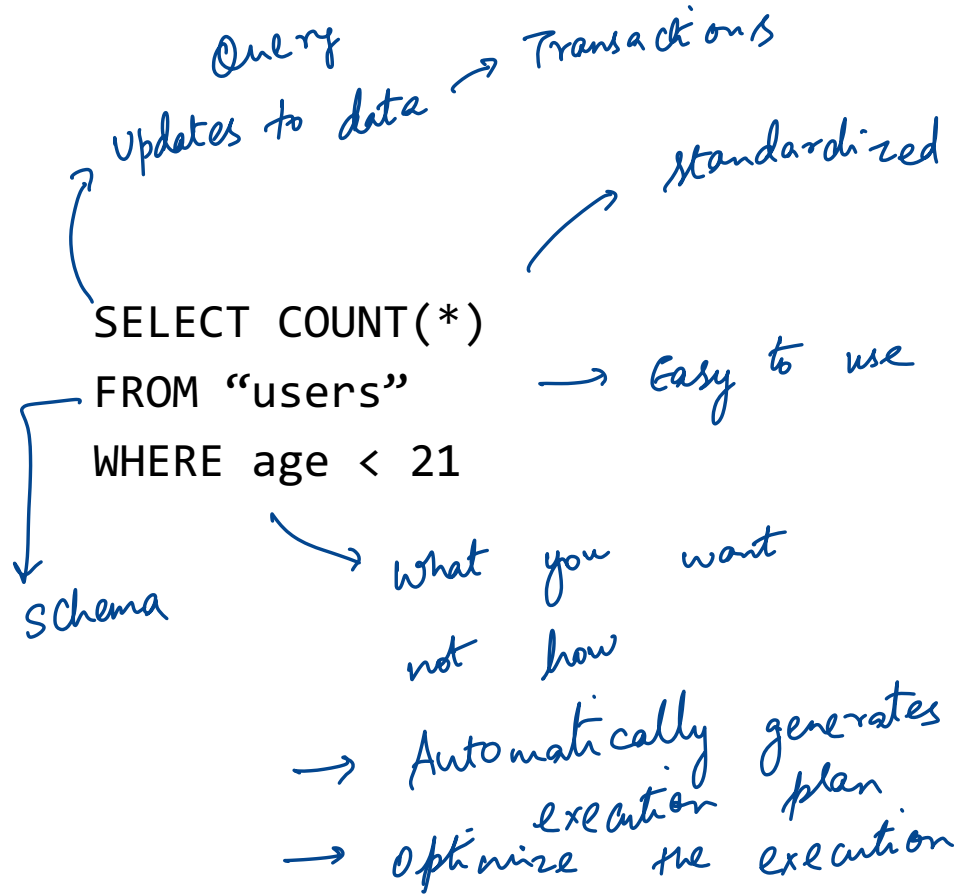
snowflake

PROCEDURAL VS. RELATIONAL

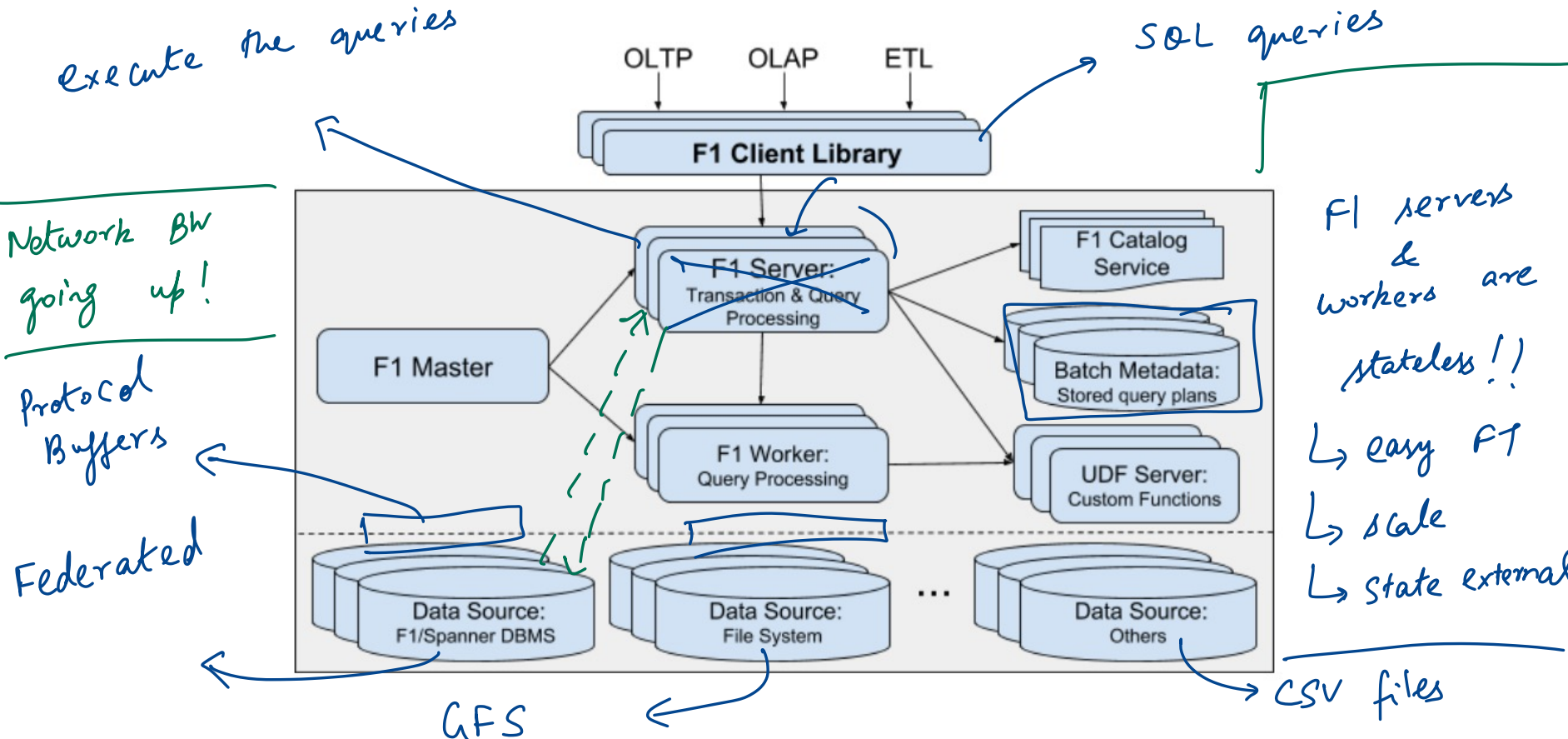
Procedural approach

```
lines = sc.textFile("users")
csv = lines.map(x =>
  x.split(','))
young = csv.filter(x =>
  x(1) < 21)
println(young.count())
```

How to execute this query



F1 QUERY DESIGN



execute the queries

SQL queries

Network BW going up!

Protocol Buffers

Federated

GFS

F1 servers & workers are stateless!!

↳ easy FT

↳ scale

↳ state external

CSV files

QUERY EXECUTION

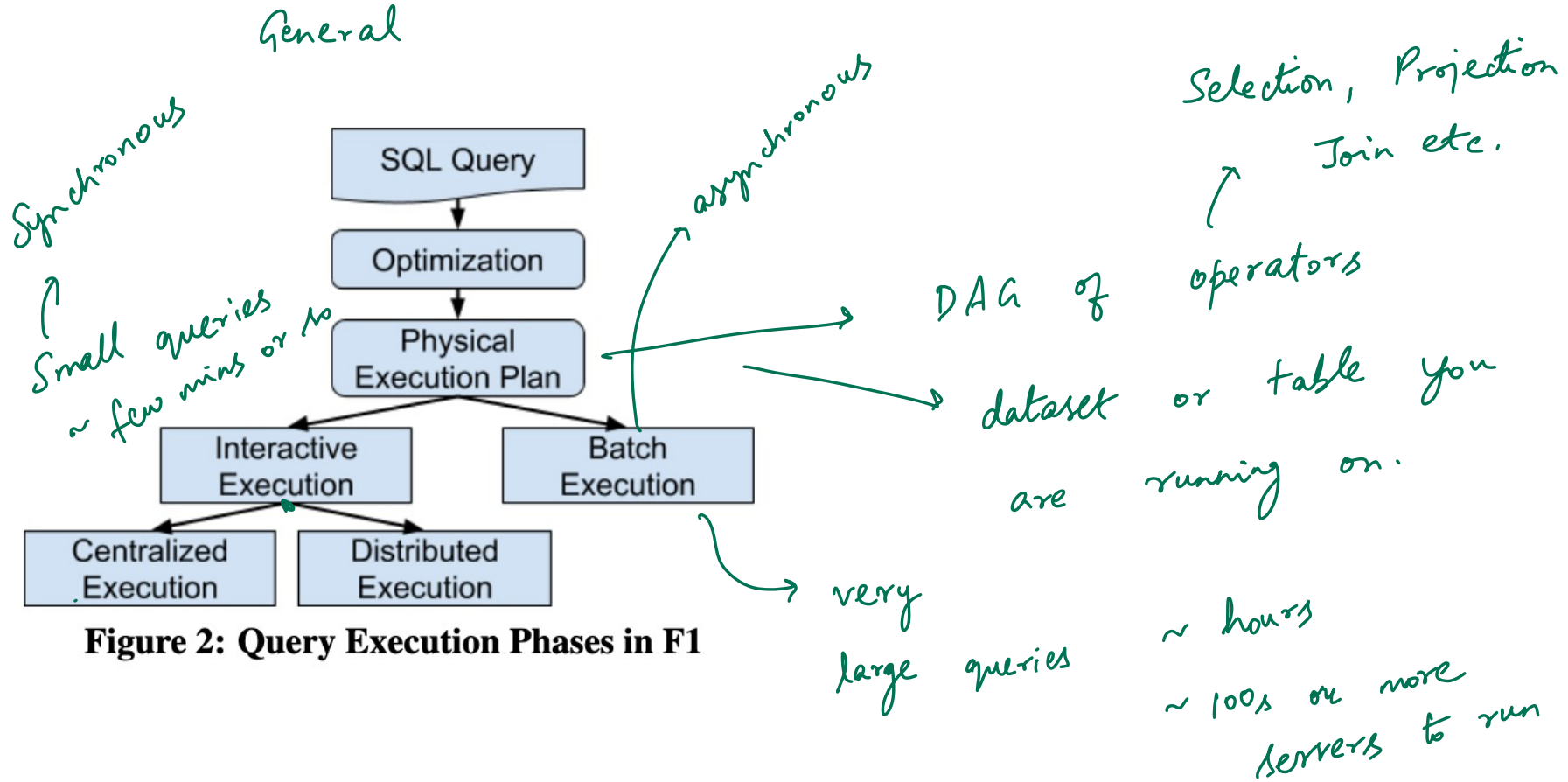


Figure 2: Query Execution Phases in F1

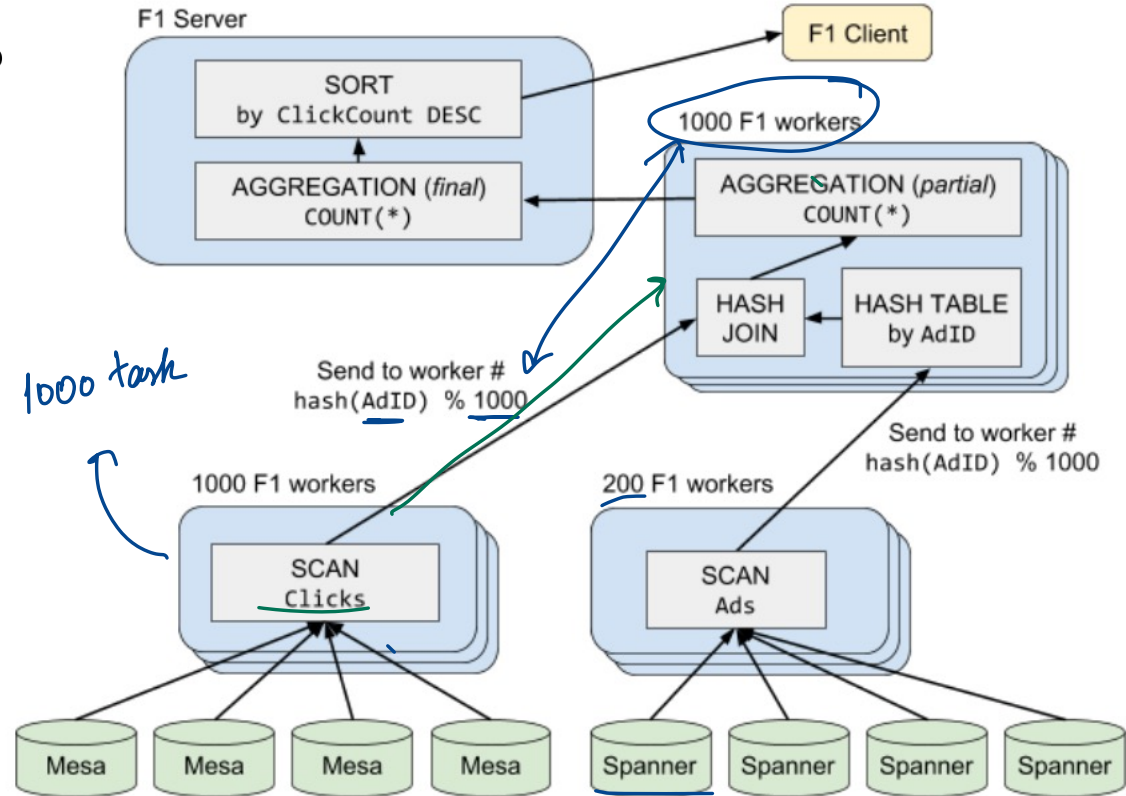
EXAMPLE

```
SELECT Clicks.Region, COUNT(*) ClickCount
FROM Ads JOIN Clicks USING (AdId)
WHERE Ads.StartDate > '2018-05-14' AND
      Clicks.OS = 'Chrome OS'
GROUP BY Clicks.Region
ORDER BY ClickCount DESC;
```

Pipelining between diff fragments
→ Don't touch disk

“shuffle”
Exchange operator? RPC based

Source sends RPC to
the dest



INTERACTIVE PERFORMANCE?

Execute in memory without check-pointing to disk!

Pipelining of query operators -- stream results (RPCs) – How?

Run complex queries in ~10ms to ~100ms

Failures? Retry query!

BATCH MODE

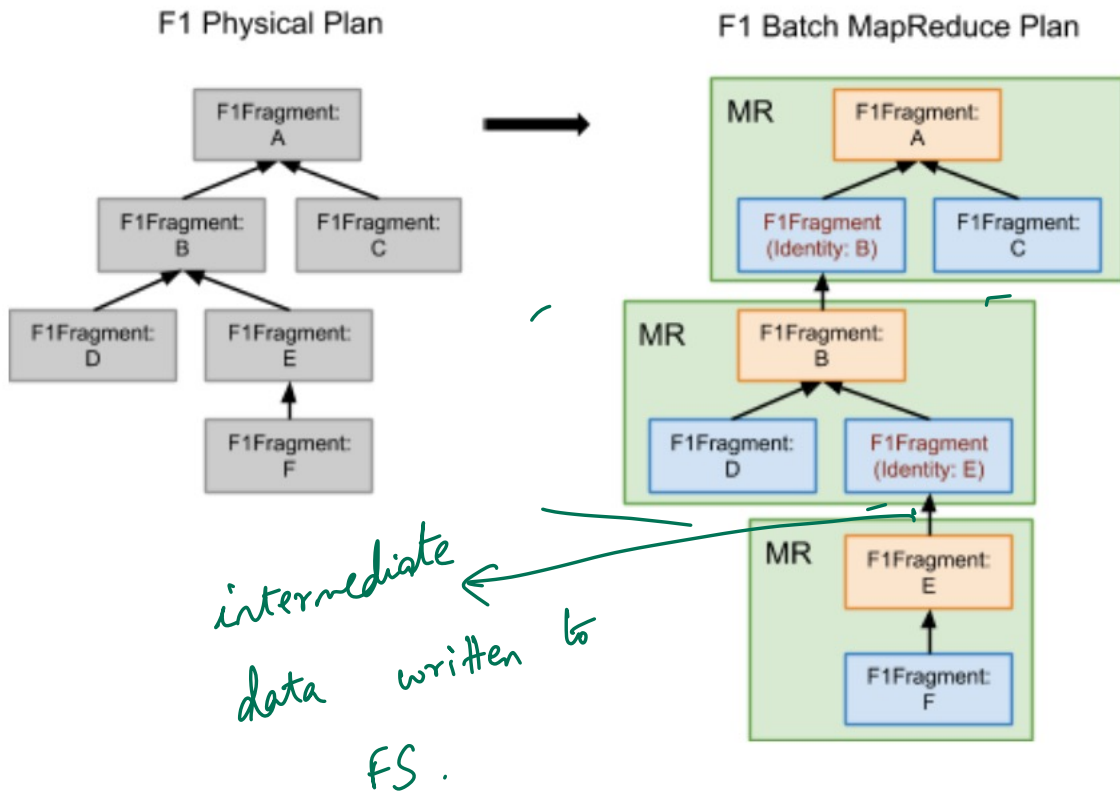
Convert manual pipelines → SQL

Benefit: query optimization

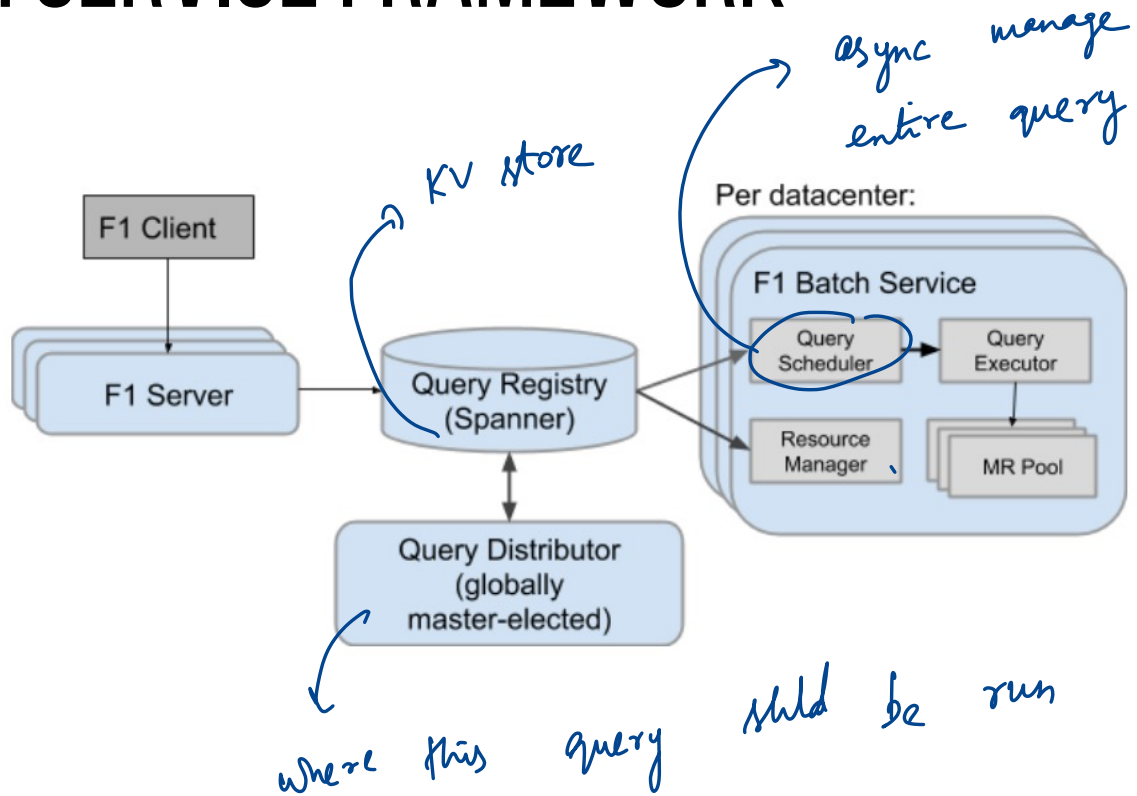
Handle server and client failures!

Approach: map each fragment to MR job

query can still execute



BATCH SERVICE FRAMEWORK



QUERY OPTIMIZER

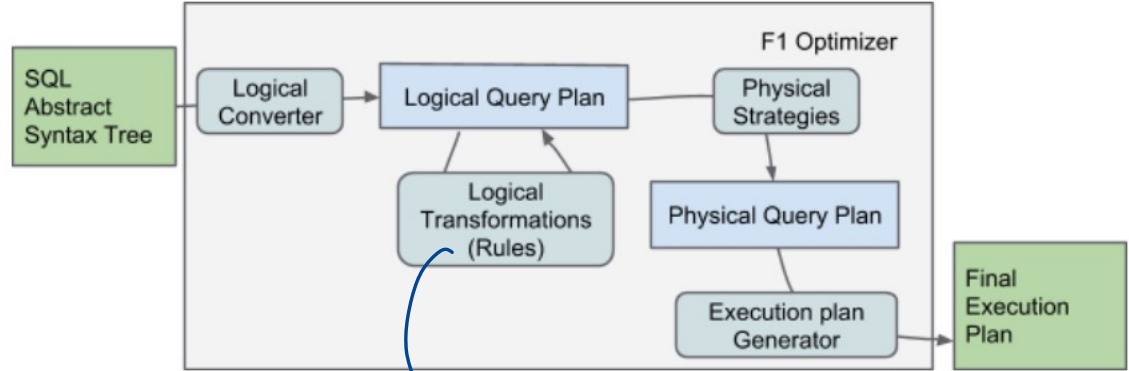
Convert query to AST

Perform passes to converge to a fixed point

Convert logical to physical

Example rules: filter pushdown, constant folding, attribute pruning, constraint propagation

→ Reduce data close to source



Optimization passes

AST → Opt pass → AST*

OTHER DESIGN DETAILS

Support for UDFs on Protocol Buffers

Robust performance operators → *avoid huge cliffs*

Columnar access? Push down / Prune away data from Protobufs at source

SUMMARY, TAKEAWAYS

Relational API

- Enables rich space of optimizations
- Easy to use, integration with C#

Scope Execution

- Compiler to check for errors, generate DAG
- Optimizer to accelerate queries (static + dynamic)

Precursor to systems like SparkSQL



<https://forms.gle/rHawi83moPMWY9L98>

DISCUSSION

Consider you have a column-oriented data layout on your storage system (Example below). What are some reasons that an FI query might be faster than running equivalent MR program?

Data layout schemes

Row Storage

Last Name	First Name	E-mail	Phone #	Street Address

0
41
81

40
80
120

Columnar Storage

Last Name	First Name	E-mail	Phone #	Street Address

0

21

41

20

40

60

metadata

→ Analyze query and only read necessary

→ Stats on Column contents, skip many rows in the scan

columns layout
 column fashion
 { 128 MB.
 1000 rows

Does FI-like Optimizer help ML workloads? Consider the code in your Assignment2.
What parts of your code would benefit and what parts would not?

NEXT STEPS

Next class: Elastic Data Warehousing with Snowflake

Project proposals due soon!

Midterm: next week