

# CS 744: F1

Shivaram Venkataraman

Spring 2025

# ADMINISTRIVIA

- Course Project Proposal: Due soon!
- Midterm details are on Piazza.

# Applications

Machine Learning

SQL

Streaming

Graph

Computational Engines

Scalable Storage Systems

Resource Management

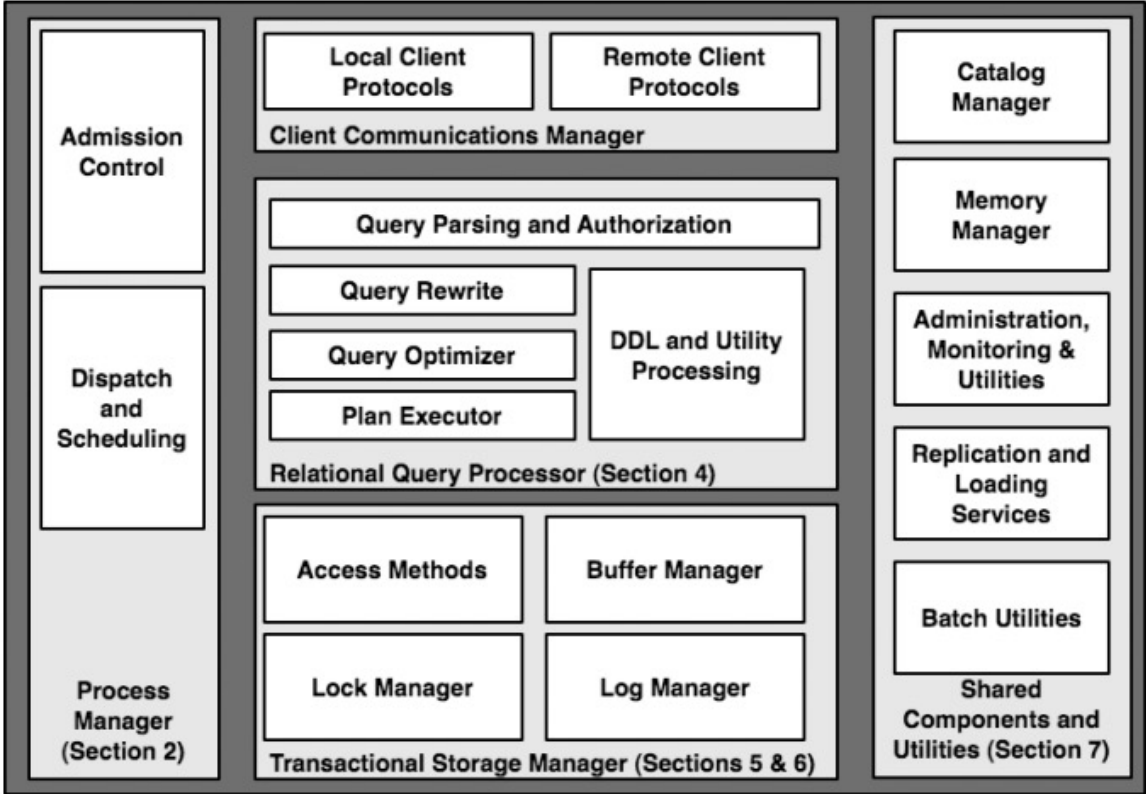


Datacenter Architecture



# SQL: STRUCTURED QUERY LANGUAGE

# DATABASE SYSTEMS

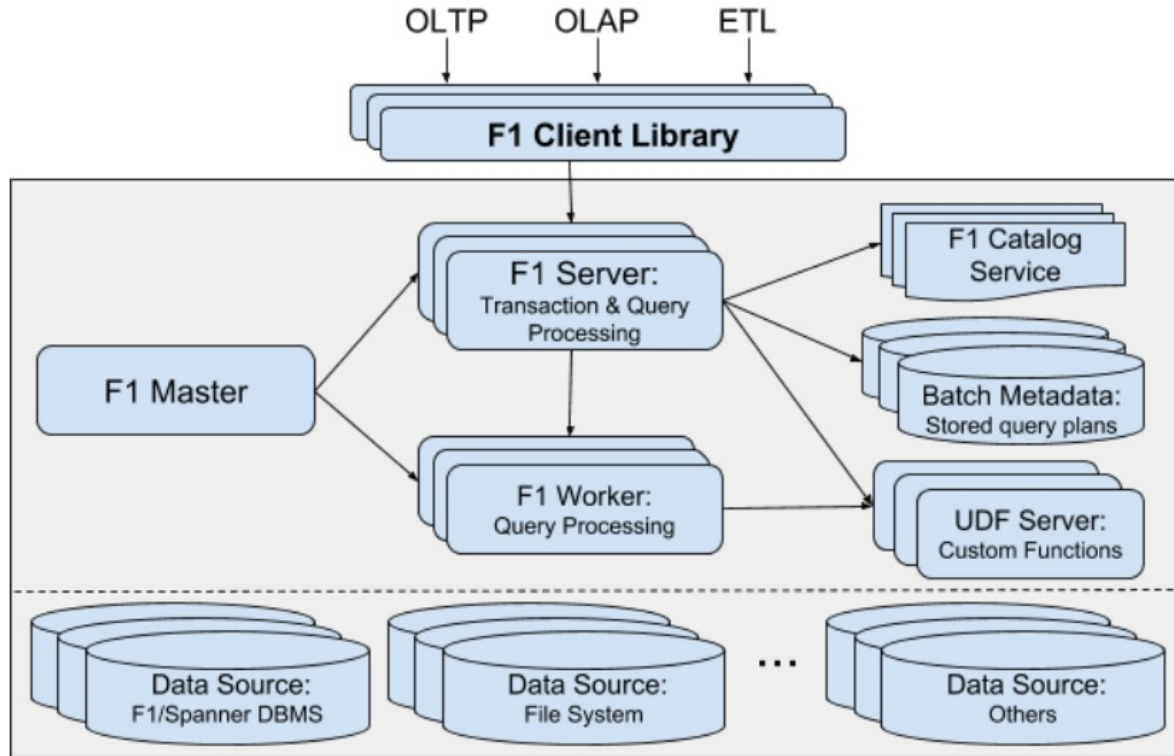


# PROCEDURAL VS. RELATIONAL

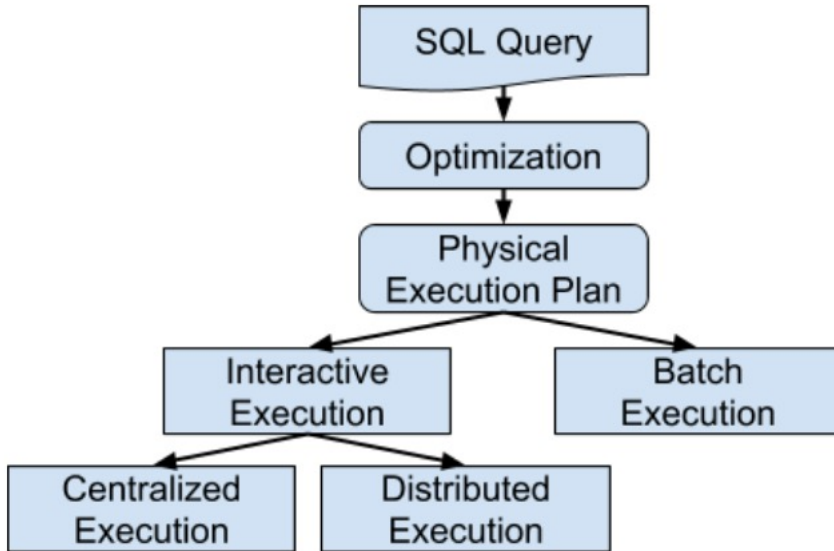
```
lines = sc.textFile("users")
csv = lines.map(x =>
    x.split(','))
young = csv.filter(x =>
    x(1) < 21)
println(young.count())
```

```
SELECT COUNT(*)
FROM "users"
WHERE age < 21
```

# F1 QUERY DESIGN



# QUERY EXECUTION



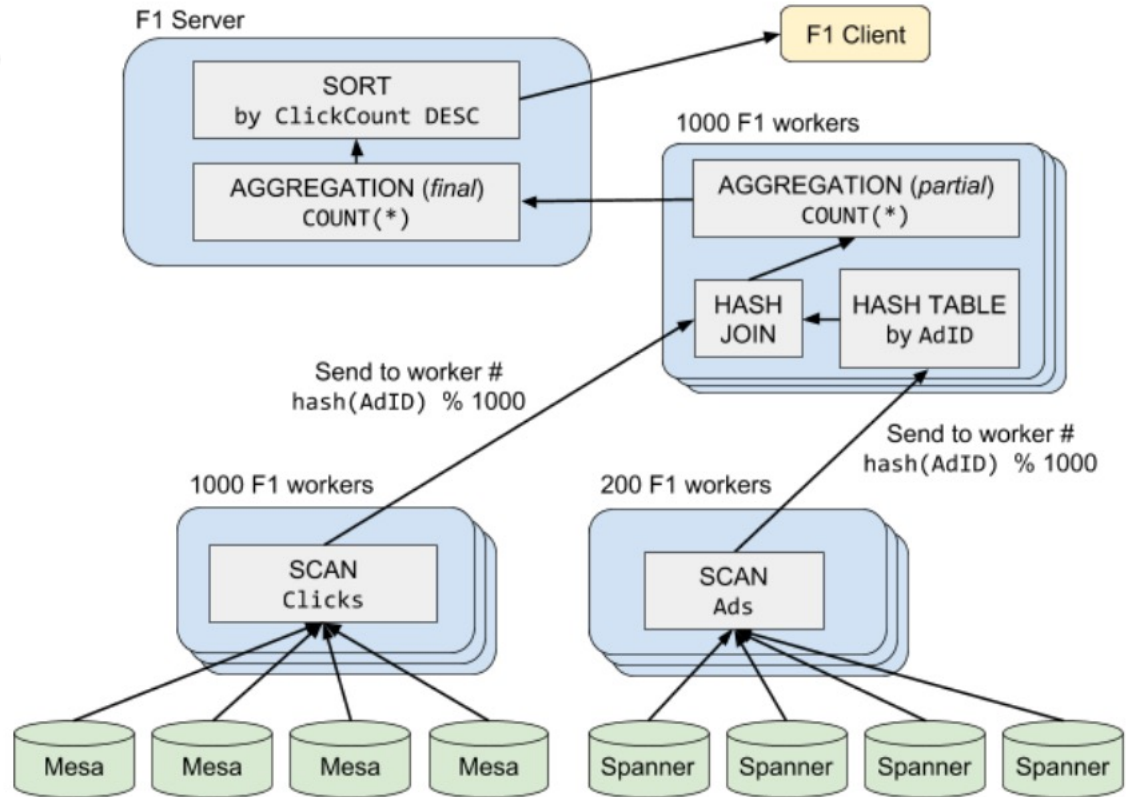
**Figure 2: Query Execution Phases in F1**



# 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;
```

Exchange operator? RPC based



# 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!

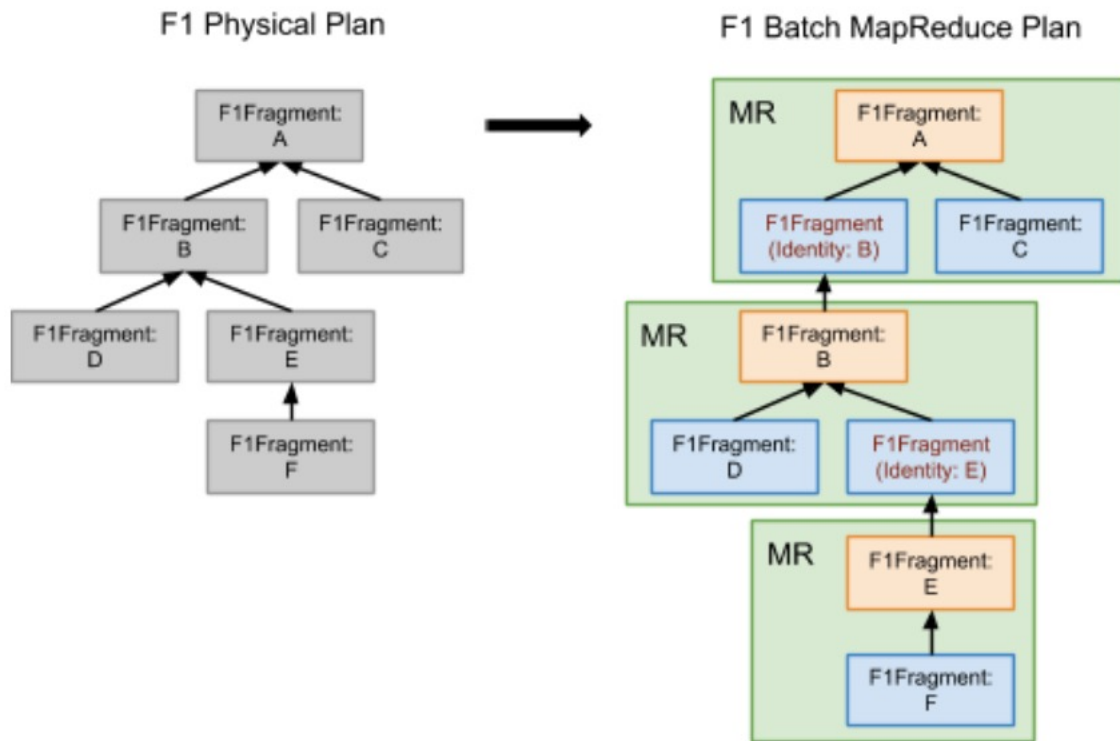
# BATCH MODE

Convert manual pipelines → SQL

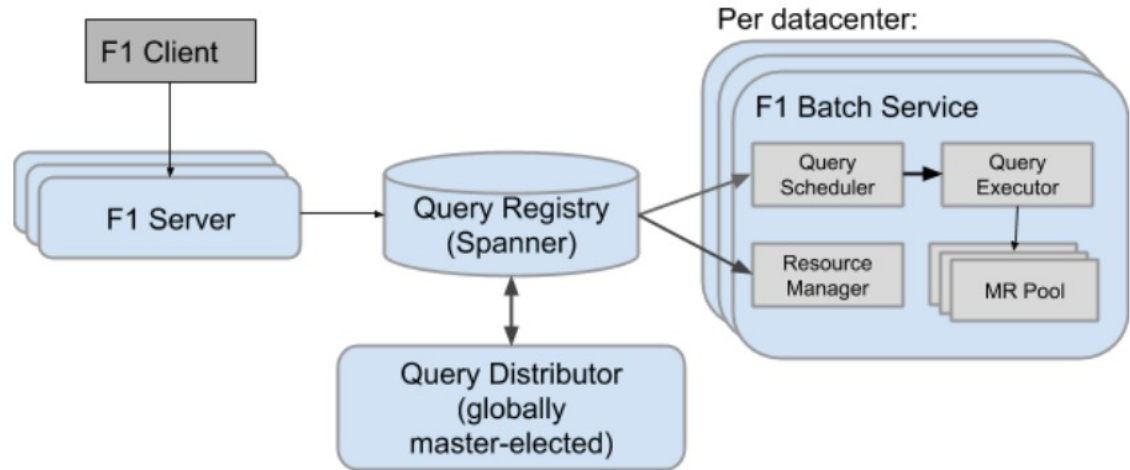
Benefit: query optimization

Handle server and client failures!

Approach: map each fragment to MR job



# BATCH SERVICE FRAMEWORK



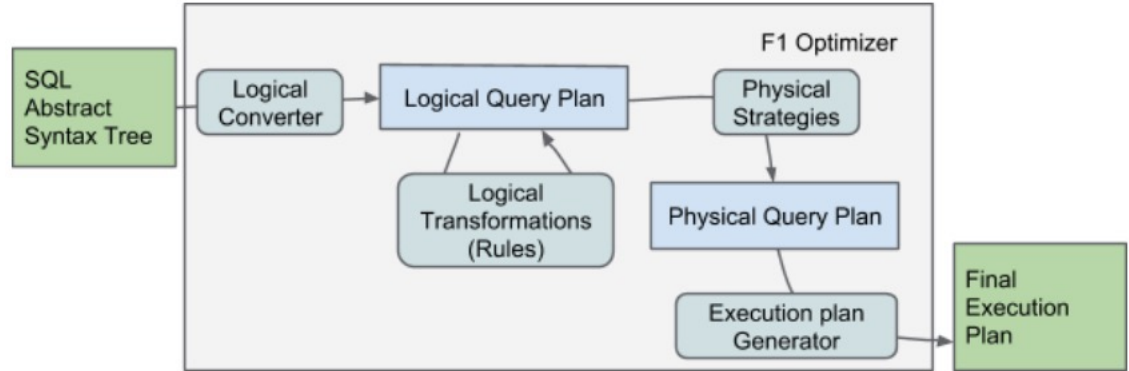
# QUERY OPTIMIZER

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



# OTHER DESIGN DETAILS

Support for UDFs on Protocol Buffers

Robust performance operators

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?

### Row Storage

Last Name	First Name	E-mail	Phone #	Street Address

### Columnar Storage

Last Name	First Name	E-mail	Phone #	Street Address

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