Good morning!

CS 744: SCOPE

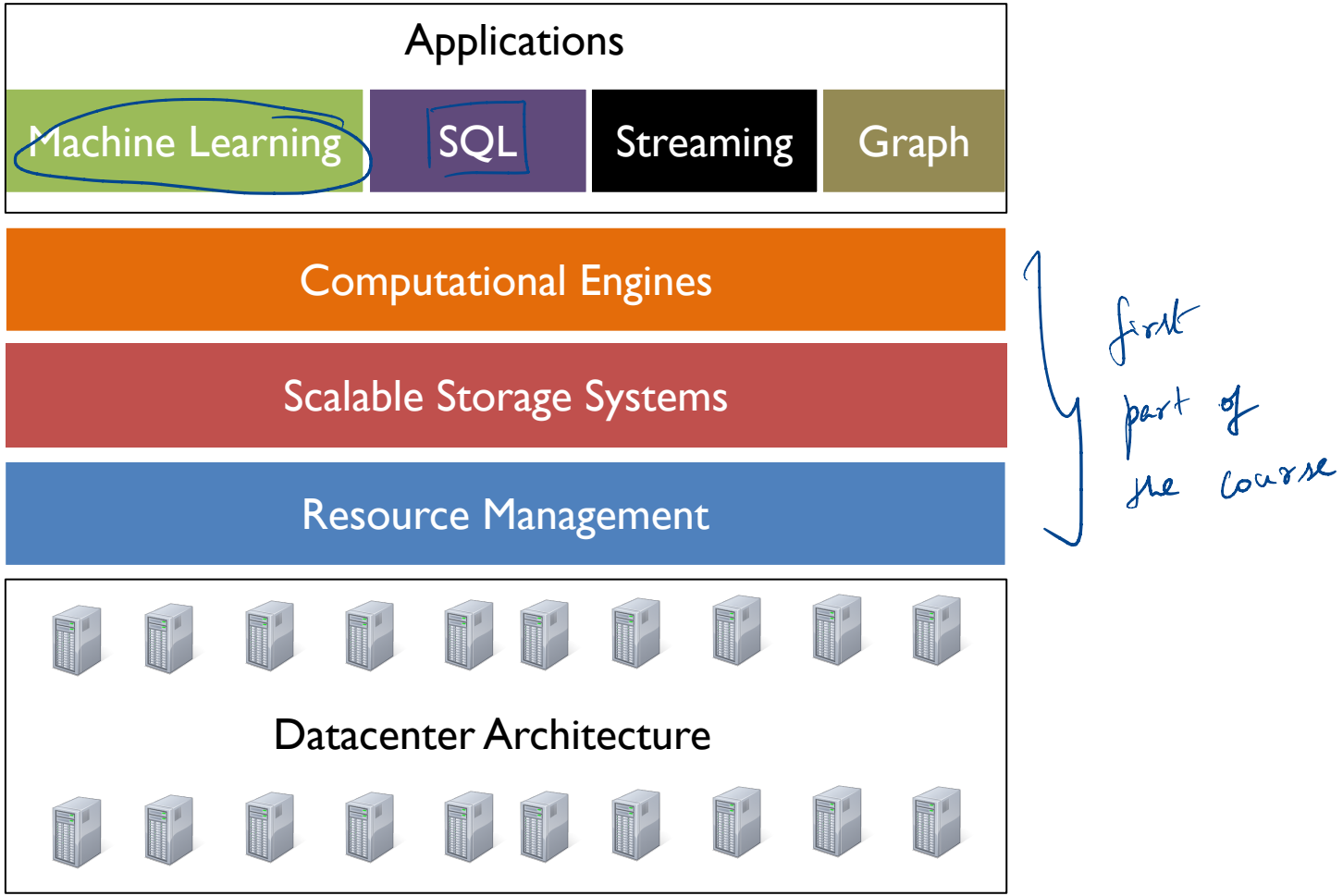
Shivaram Venkataraman

Fall 2021

ADMINISTRIVIA

- Assignment 1 grades: this week → Today
- Course Project Proposal: Due Monday → PDF file
  - 1 - 3 pages maximum excluding referen
- Midterm more details today on Piazza

Systems for ML



first part of the course

SQL: STRUCTURED QUERY LANGUAGE

DATABASE SYSTEMS

Relational Databases

↳ Query language

SQL

submit to

Database

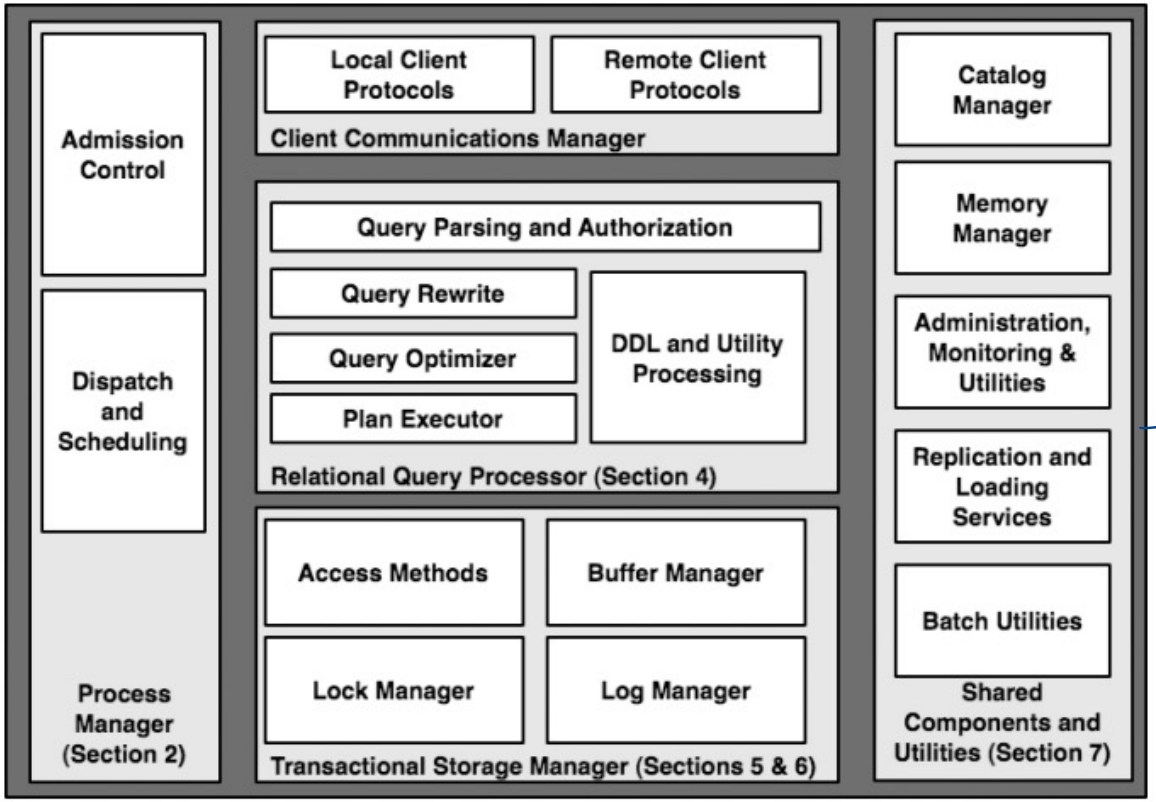
Analytic Queries (OLAP)

↳ Batch queries, on large data

Results

OLTP or Transactional Queries

~ low latency online



PROCEDURAL VS. RELATIONAL

Flexibility, user defined operations for splitting etc.

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

Parsing

intermediate variables can be re-used?

How to execute this query

more intuitive

What you want to execute

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

Column name

"schema"

Microsoft

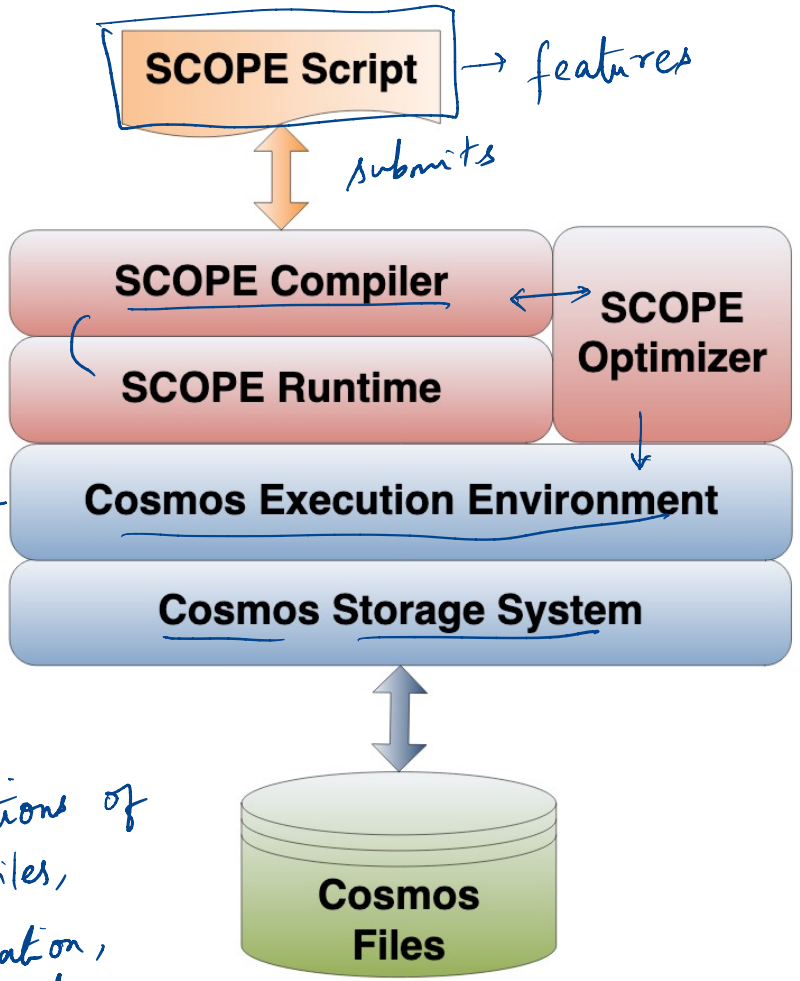
SCOPE

```
SELECT query, COUNT(*)
AS count
FROM "search.log"

USING LogExtractor
GROUP BY query
HAVING count > 1000
ORDER BY count DESC;
```

Dryad
Generalization of MR
DAG of operators

GFS
partitions of files, replication, append



SCOPE OPERATORS

Input reading: What is different?

Extractors to read inputs from Cosmos storage

```
EXTRACT column[:<type>][, ...]
FROM <input_stream(s) >
USING <Extractor> [(args)]
[HAVING <predicate>]
```

Schema of the "rowset" (RDD)

file

filter applied when you read data

CSVExtractor, ... user defined Extractors → flexibility for diff data formats

SQL OPERATORS

Select – read rows that satisfy some predicate

Join – Equijoin with support for Inner and Outer join

↳ Subset of joins found in SQL standard

GroupBy – Group by some column

OrderBy – Sorting the output

Aggregations – COUNT, SUM, MAX etc.

Standard SQL operators

→ familiarity for users who already know SQL

LANGUAGE INTEGRATION

UDFs → Databases

R = EXTRACT ... <file>

C# function that operates on strings

```
R1 = SELECT A+C AS ac, B.Trim() AS B1
FROM R
WHERE StringOccurs(C,"xyz") > 2
```

→ invoke the executable

```
#CS
public static int StringOccurs(string str, string ptrn){
 int cnt=0; int pos=-1;
 while (pos+1 < str.Length) {
    pos = str.IndexOf(ptrn, pos+1);
    if (pos < 0) break;
    cnt++;
  }
  return cnt;
}
#ENDCS
```

← pre-processor guide

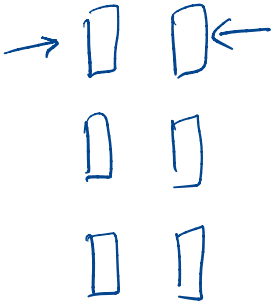
→ UDF is a black box, Optimizer doesn't know about it!

← C# function definitions

UDF is the programmers responsibility!

Compile this / generate exec

MAPREDUCE-LIKE?



Process → Very similar to Map

Input rowset → Output rowset

schema needs to be specified!

SQL GROUP BY

Reduce → Operates on grouped data

Reduction function is called once per group

Combine

```
COMBINE S1 WITH S2
ON S1.A==S2.A AND S1.B==S2.B AND S1.C==S2.C
USING MultiSetDifference
PRODUCE A, B, C
```

2 rowsets which are "co-partitioned"

EXECUTION: COMPILER

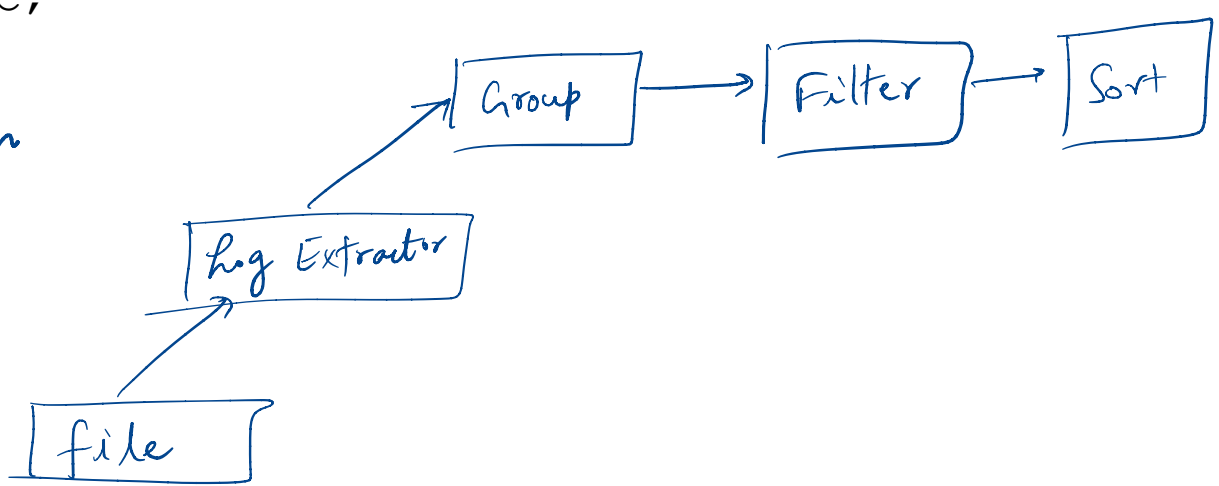
```
SELECT query, COUNT() AS count
FROM "search.log"
USING LogExtractor
GROUP BY query
HAVING count > 1000
ORDER BY count DESC;
```

Check syntax, resolve names

Checks if columns have been defined

Result: Internal parse tree

Logical query plan



OPTIMIZER

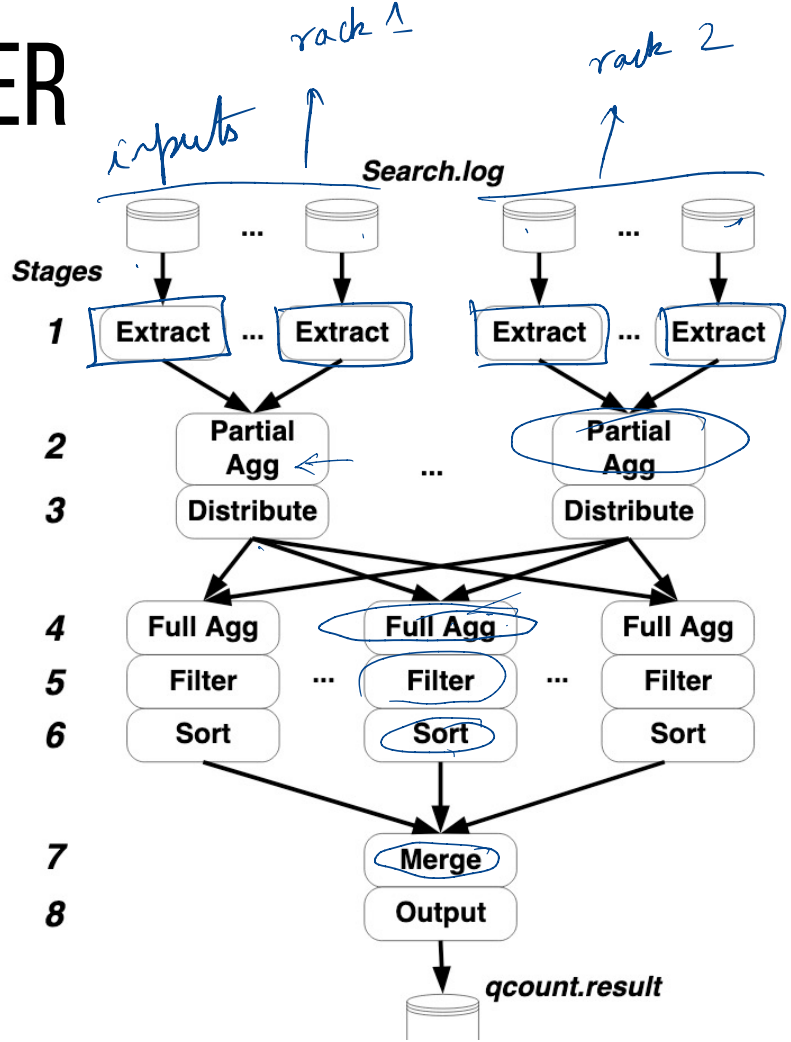
Rewrite the query expression → lowest cost

Examples:

- Removing unnecessary columns
- Pushing down selection predicates
- Pre-aggregating or partial aggregation

rule based

```
SELECT query, COUNT() AS count
FROM "search.log"
USING LogExtractor
GROUP BY query
HAVING count > 1000
ORDER BY count DESC;
```



RUNTIME OPTIMIZATIONS

Hierarchical aggregation

↳ Similar to query tree in prev. slide

↳ Insert rack-level partial aggregations to minimize network

Locality-sensitive task placement

↳ Spark / MR, which is to place a task close to where its inputs are

Simple to do for extractors

also possible to do this for other operators

SUMMARY, TAKEAWAYS

Relational API ← Schema

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

Scope Execution

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

Precursor to systems like SparkSQL, Apache Hive, ...

DISCUSSION

https://forms.gle/eaCacDp6budf5cTDA

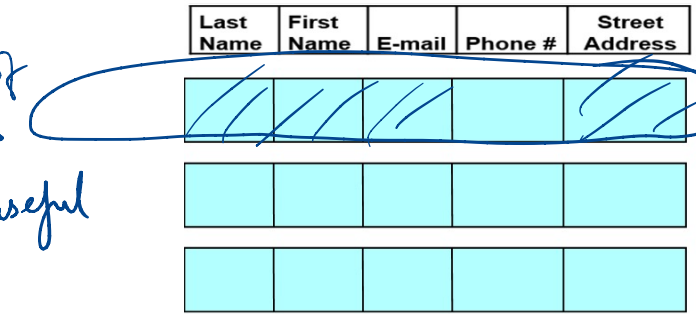
Consider you have a column-oriented data layout on your storage system (Example below). What are some reasons that a SCOPE query might be faster than running equivalent MR program?

random access to get 1 col.

Sequential access 1 col.

Translate column storage to row based API

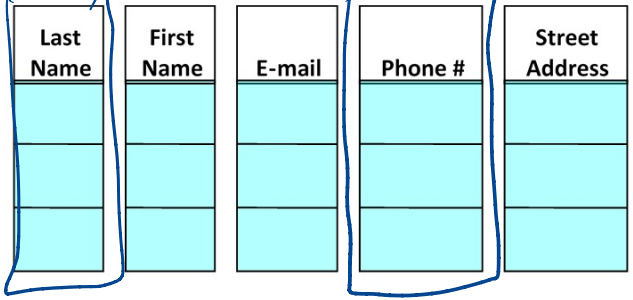
Row Storage



80% of this is not useful

read in entire row
- extract phone #
throw away rest of it

Columnar Storage



SCOPE can filter better?
- query only touches few columns
⇒ SCOPE can only read those columns from storage

http://dbmsmusings.blogspot.com/2017/10/apache-arrow-vs-parquet-and-orc-do-we.html

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

ML pipelines do featurization

↳ Extractor, only reading data that you need

Hierarchical aggregation

↳ All Reduce → automatic partial agg. at rack level

Deep learning operators → Convolution, ReLU

↳ Relational Optimizer all of this is a UDF

↳ Deep Learning optimizer operators in Conv, ReLU ....

NEXT STEPS

Next class: Elastic Data Warehousing with SnowFlake

Project proposals due Monday! See Piazza!

Midterm coming up next week