CS 744: SCOPE

Shivaram Venkataraman
Fall 2022
- Assignment 1 grades out!
- Course Project Proposal: Due soon!
  - Google Cloud Credits ( $150 )
- Midterm details are on Piazza. 1pm to 2.15pm CS 1221. Oct 27th
- No reviews for Tuesday (Snowflake)!
Scalable Storage Systems

Datacenter Architecture

Applications

- Machine Learning
- SQL
- Streaming
- Graph

Computational Engines

Resource Management

Training
Scheduling
Inference

SCOPE
Spark SQL
Cloud-based
Snowflake
SQL: STRUCTURED QUERY LANGUAGE
DATABASE SYSTEMS

Relational Model

Query

Declarative query

submit

Process Manager (Section 2)

Admission Control

Dispatch and Scheduling

Client Communications Manager

Local Client Protocols

Remote Client Protocols

Query Parsing and Authorization

Query Rewrite

Query Optimizer

Plan Executor

Relational Query Processor (Section 4)

Access Methods

Buffer Manager

Lock Manager

Log Manager

Transactional Storage Manager (Sections 5 & 6)

DDL and Utility Processing

Catalog Manager

Memory Manager

Administration, Monitoring & Utilities

Replication and Loading Services

Batch Utilities

Shared Components and Utilities (Section 7)

Processor

Query

Tuples or rows
PROCEDURAL VS. RELATIONAL

- Programming Style  Re-use of data
- Flexibility       Custom functions

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
println(young.count())

Declarative \rightarrow "What you want"
System figures out \rightarrow "How"

"How" you want to do this

\downarrow lowers barrier to entry
Example

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

Input reading: What is different?

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

Select – read rows that satisfy some predicate
Join – Support for Inner and Outer join

GroupBy – Group by some column
OrderBy – Sorting the output
Aggregations – COUNT, SUM, MAX etc.

Subset of the SQL standard
Familiar for users
- easy to use
**LANGUAGE INTEGRATION**

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

#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
**MAPREDUCE-LIKE?** → Mix and match

**Process (Map)**
- Input rows given to a user defined class → One or more rows output (with schema)

**Reduce**
- grouped data All rows that belong to a group → One or more rows output

**Combine**
- COMBINE S1 WITH S2
- USING MultiSetDifference
- PRODUCE A, B, C

→ Take two tables which are co-partitioned → One table output

**NOT SIMILAR TO MR COMBINE**
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
- Check if column is a number if + on it
- Logical query plan: LogExtractor → Select → Group → Order
Rewrite the query expression → lowest cost

Examples:
- Removing unnecessary columns
- Pushing down selection predicates
- Pre-aggregating

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

Hierarchical aggregation

- Partial agg. and full aggregation run time
- Insert it based on need

Locality-sensitive task placement

- Place task close to its inputs
- Change the query plan at
- Job Manager locality hint
- Execution where are tasks running
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, Apache Hive
DISCUSSION

https://forms.gle/CCx6evn2LJAY8m9d9
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?

Row Storage

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>E-mail</th>
<th>Phone #</th>
<th>Street Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Columnnar Storage

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>E-mail</th>
<th>Phone #</th>
<th>Street Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

→ map stage extract one column, then aggregation

query only touches one column

→ push down selection to only read 1 column from disk!

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?

Within 1 iter:

- Pick between All Reduce vs. Gather (Scatter)
- Insert rack or machine-level partial agg
- Optimizer needs to be across-iterations
- Pipeline I/O with compute
- Forward + Backward pass?
- Improve input reading for ML workloads
- I/O reduction
Next class: Elastic Data Warehousing with SnowFlake
Project proposals due soon! See Piazza!
Midterm: next week