CS 744: GEODE

Shivaram Venkataraman

Fall 2019

ADMINISTRIVIA

- Assignment 2 grades
- Midterm coming up Tuesday!
- AEFIS feedback form

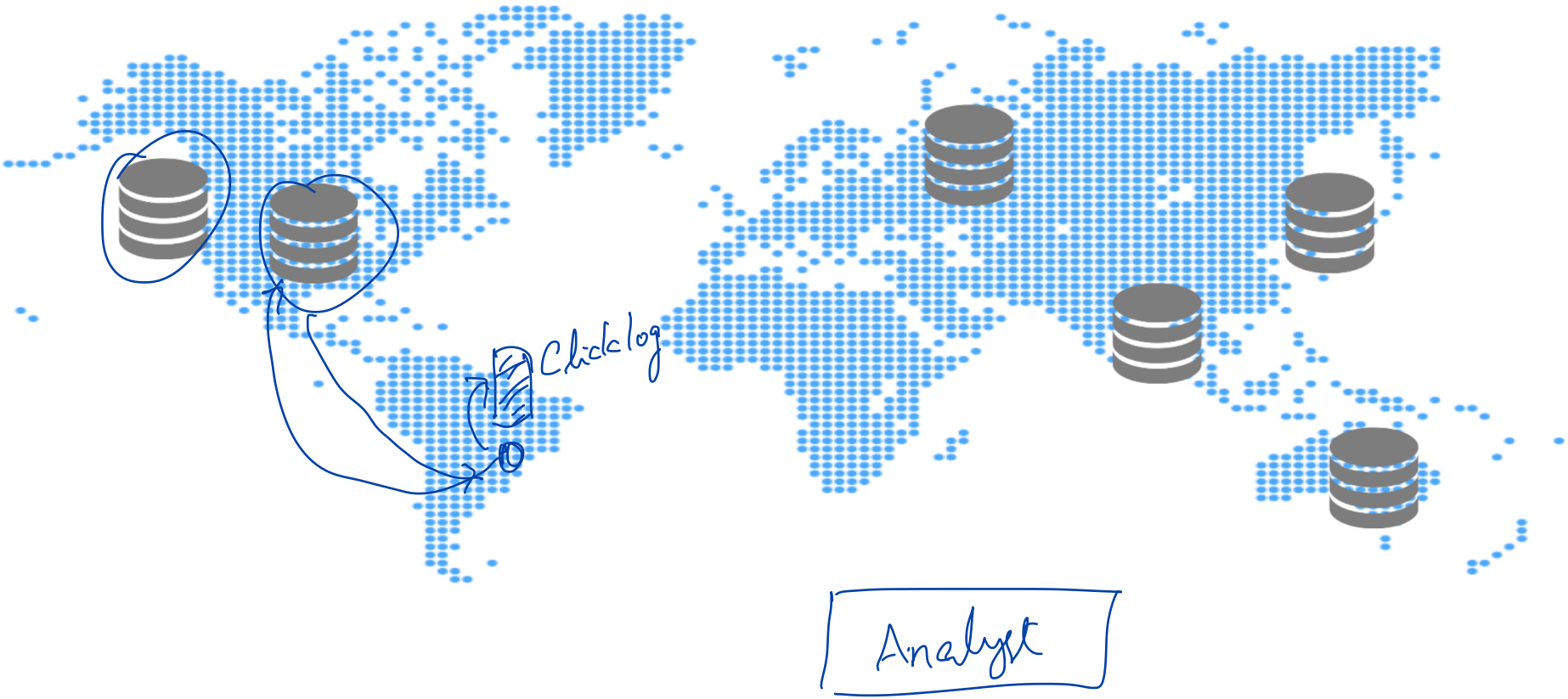
SQL IN BIG DATA SYSTEMS

- Scale: How do we handle large datasets, clusters ? → Spark SQL

- Wide-area: How do we handle queries across datacenters ?

WIDE AREA ANALYTICS

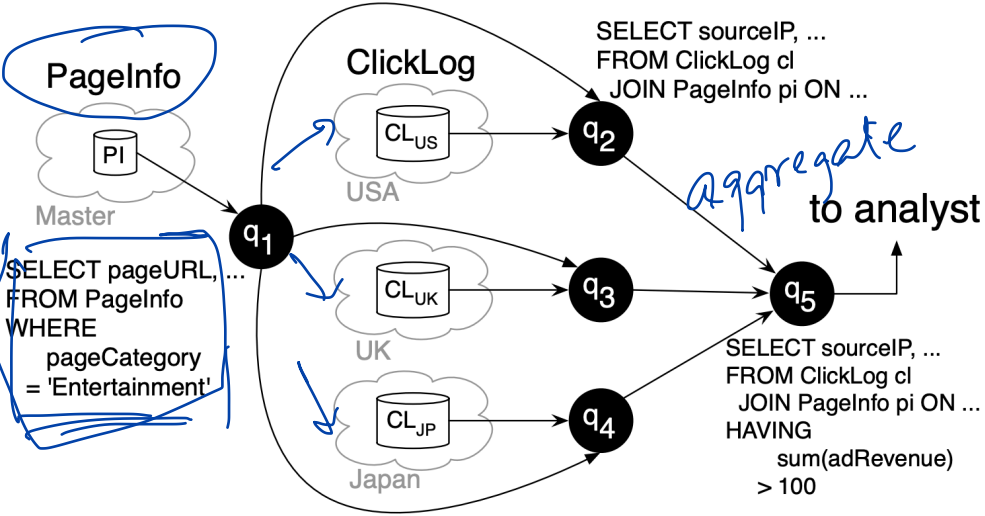
MOTIVATION



GOALS / ASSUMPTIONS

- Support analytics queries (including joins)
- Minimize wide-area network usage ← Across datacenter links. low bandwidth
- Resources within single DC are plentiful → Not optimized for
- Primary metric: Bandwidth cost not latency

EXAMPLE




```
Q: SELECT sourceIP, sum(adRevenue), avg(pageRank)
   FROM ClickLog cl JOIN PageInfo pi
     ON cl.destURL = pi.pageURL
   WHERE pi.pageCategory = 'Entertainment'
   GROUP BY sourceIP
   HAVING sum(adRevenue) >= 100
```

Sizes of the table

↳ Join How do we do it

Broadcast

↳ Filter before Bcast

↳ Do a local join

↳ Aggregation, return results

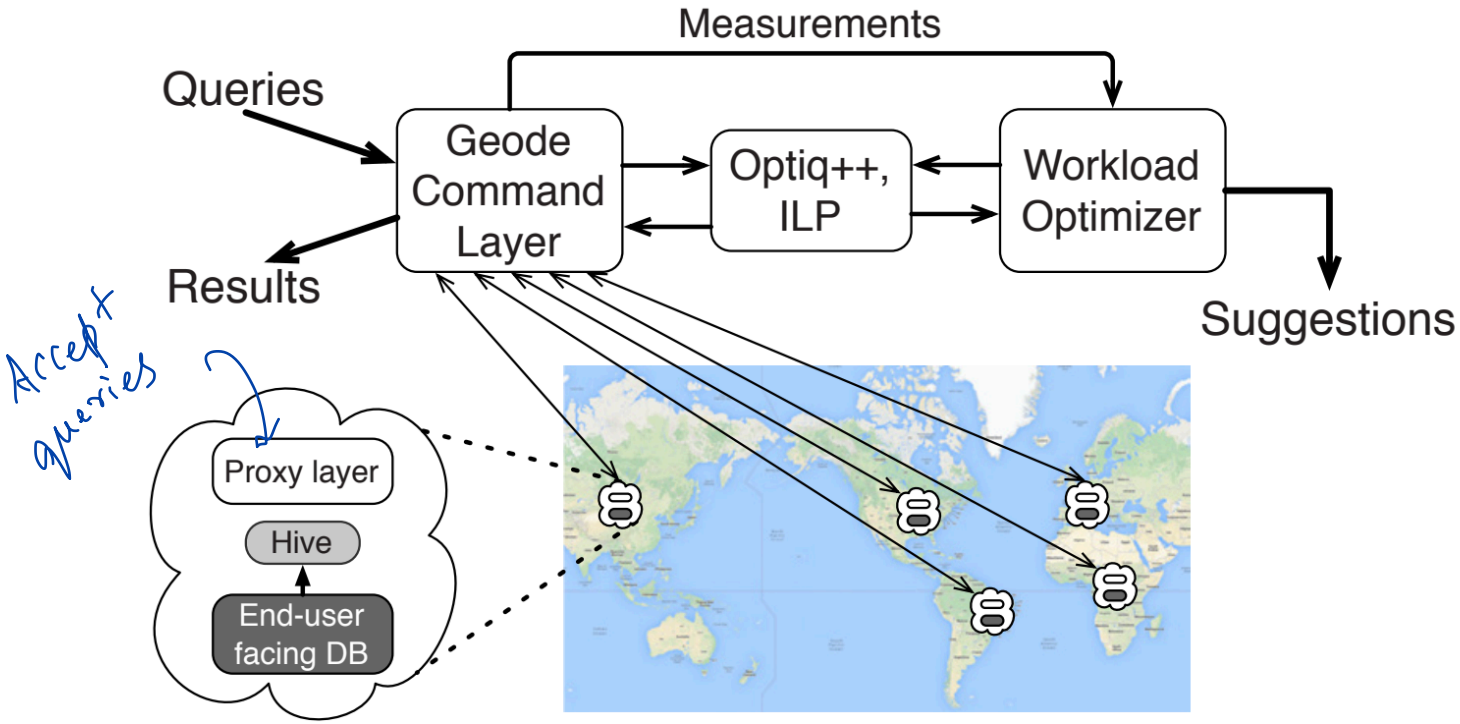
APPROACH

1. Join order selection
   - Choice of join algorithm
   - Order in which they are executed

2. Task assignment

3. Manage data replication

ARCHITECTURE



OPTIMIZER SETUP

Datacenters / topology slowly evolving

Workload properties

Data birth

Produced by other process

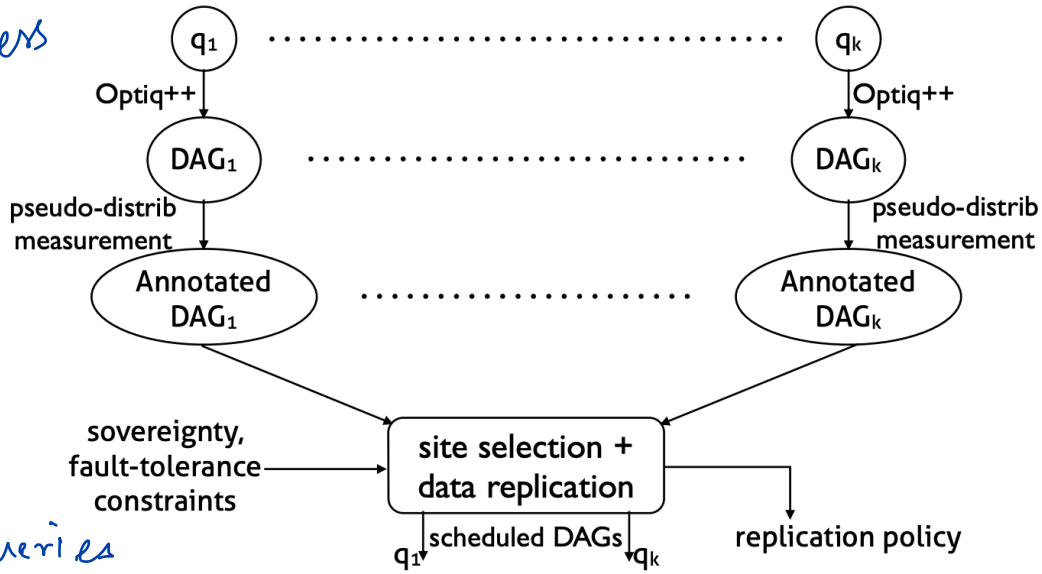
Sovereignty : Restriction on data movement

Fixed Queries

↳ Does data change?

↳ Small number of queries

Queries evolve very slowly



SUB QUERY DELTAS

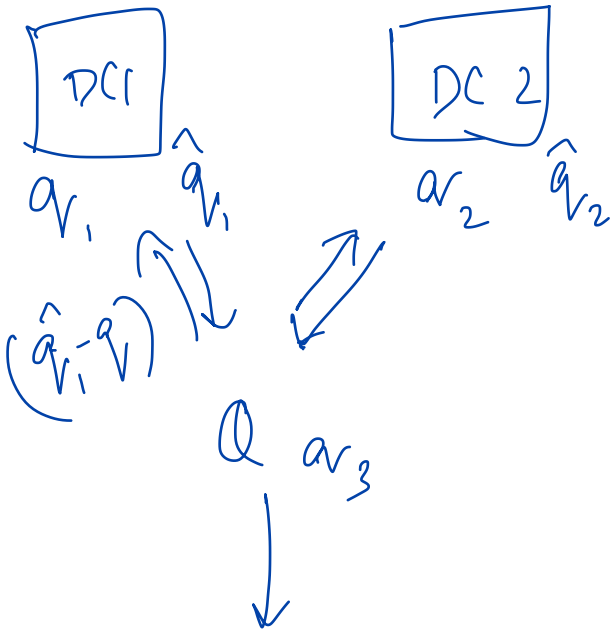
Cache intermediate results in sub-queries

What does this help ?

- Repeated queries (issued every hour etc.)
- Shared sub-queries (across data-scientists ?)

What does this not help with?

- Computation still happens within DC
- Extra storage for cache (how do you expire this ?)



QUERY OPTIMIZER: CALCITE++

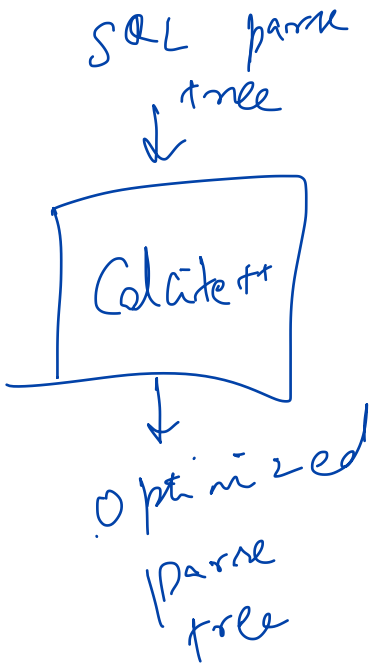
Apache Calcite: centralized SQL query planner

Input: SQL parse tree. Output: Optimized parse tree

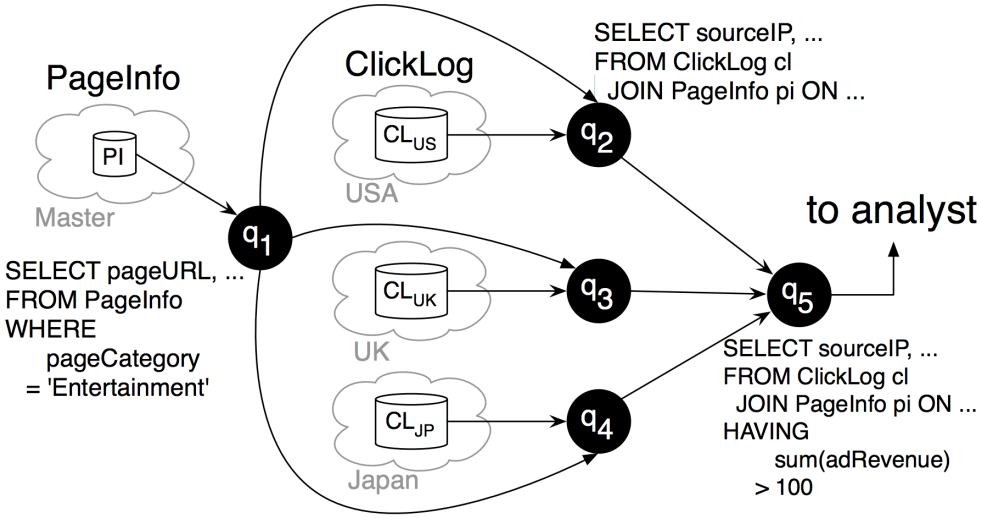
Similar to Catalyst, but includes cost-based optimization

Calcite++

- Estimate distributed join cost
- Important to pick right plan not estimate accurate cost!
- Select join strategy e.g. Broadcast, 2 more?

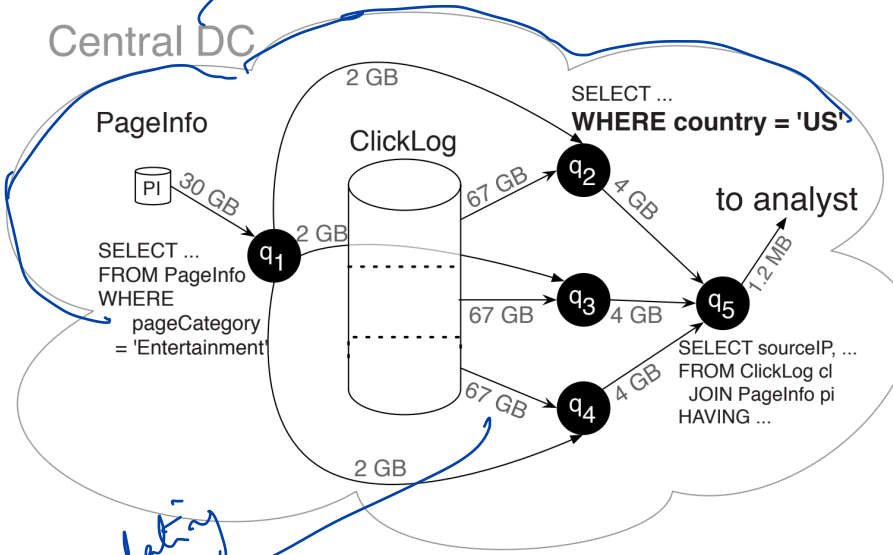


PSEUDO DISTRIBUTED EXECUTION



Original

Actually running in 1 DC



Simulating effect of multiple DC

Pseudo Distributed

PSEUDO DISTRIBUTED EXECUTION

Key idea: Use stats from repeated executions

Advantages

Precise estimation

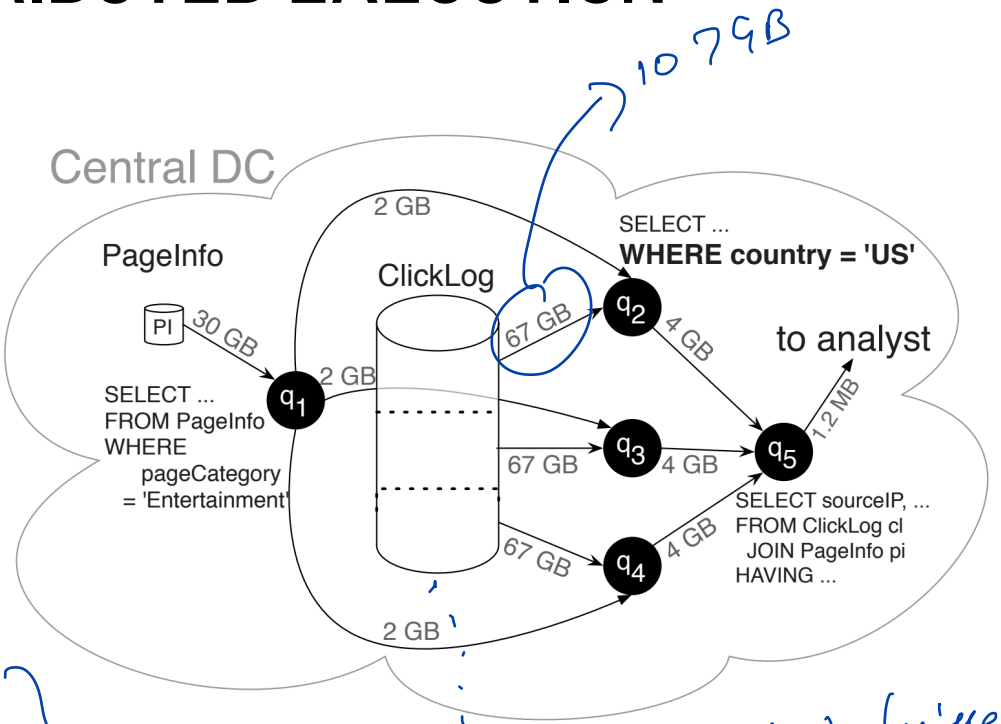
General across operators

Disadvantages ?

Overhead of fetching execution

Model fluctuations in Network, Data

Space of executions is large



Latency

Caching hits/misses

SITE SELECTION, DATA REPLICATION

Integer linear program formulation

Objective: Minimize replicationCost + executionCost

where the computation is scheduled to run.

Constraints

- Disaster recovery
- Regulatory constraints

where the data is allowed to be stored

Solution

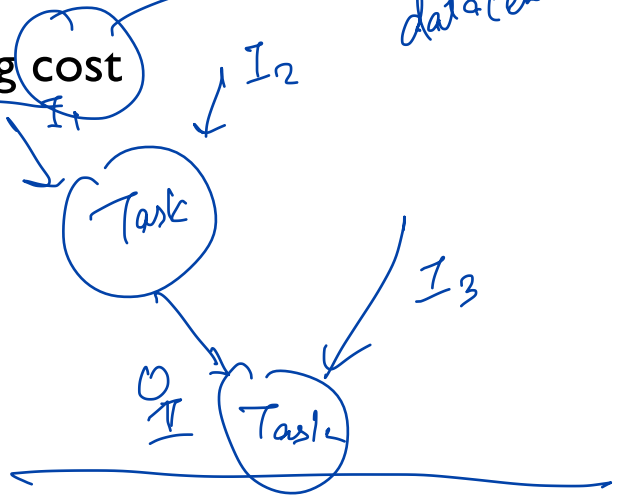
- Assignment of which task runs on which DC
- Which partition is replicated to which DC

SITE SELECTION, DATA REPLICATION

ILP doesn't scale for large workloads → either large number of queries or DCs

Greedy heuristic

- Greedily pick datacenter for task based on copying cost → BW across datacenter
- Plugin values, run ILP for replication strategy



Limitations

→ Solutions may not be optimal

→ Fixed workload

Cost model, caching etc.

SUMMARY

New area of wide-area big data analytics

Combine query optimization + network awareness

Main contributions

- Optimize data replication, task placement
- Intelligent caching to reuse sub-queries

AEFIS survey!

DISCUSSION

https://forms.gle/Qr142WN1LVNyVAfLA

```
Items(id: Int, name: String, price: Double)
Orders(id: Int, itemId: Int, count: Int, loc: String)

SELECT order.id, item.name, item.price, order.count
FROM item
JOIN order
WHERE item.id = order.itemid and item.price < 1400 and order.count > 2 - 1
```

Caching: Orders are monotonic
→ Will help a lot!

If the orders table was distributed across three geographic locations: US, Europe and Asia, how can the query can be executed by using Geode.

Depends on sizes of tables.

↳ Filter first on item price, order count

↳ Do either centralized join or broadcast items & join.