# Paper Summary

**"Global analytics in the face of bandwidth and regulatory constraints"**

**"Clarinet: WAN-Aware Optimization for Analytics Queries "**

Boyu Tian (bytian), Rui Liu (ruixliu), and Haojun Ma (mahaojun)

## Problem and Motivation

As the data becomes larger and larger, geographically distributed data centers are deployed by global-scale companies, like Google, Facebook and so on.To analyze and query such data that are distributed among multiple data centers, most existing systems pull all data into a central data center and run analytics program within the central data center. However, transferring large-scale data among Wide-Area Network (WAN) links is very expensive because of the bandwidth limit of transoceanic links. Some emerging regulatory constraints may even render such transferring impossible, due to privacy issues. This paper tried to tackle the problem of Wide-Area Big Data (WABD) which consists in orchestrating query execution across data centers to minimize data transfer while respecting regulatory constraints. Specifically, they combine classical query planning with WAN-aware mechanisms such as deltas on cached subquery results, pseudo-distributed execution and joint query optimization.

## Hypothesis

- Resources within a single data center (such as CPU and storage) are relatively cheap compared to cross-data center bandwidth.
- There are some regulatory constraints on the raw data movement but the derived information, such as aggregates, models and reports can still cross geographical boundaries.
- Large-scale data is stored across geo-distributed data centers, which evolves smoothly and slowly.
- Analysis is expressed in SQL-like queries, which will often recur.
- Cost of data transfer when supporting analytics execution is the primary metric of interest, rather than the execution latency.

## Solution Overview

They optimize the execution strategy to minimize bandwidth and respect sovereignty. Specifically, they make the following contributions:

1. Query planning: they extend Apache Calcite to output query execution plan that is optimized in the geo-distributed setting.
2. Pseudo-distributed measurement: they develop a technique to modify query in order to measure the data transfer volume along each edge of DAG corresponding to the query.
3. Site Selection and Data Replication: they jointly optimize site selection and data replication to minimize bandwidth cost. This is modeled as an Integer Linear Programming (ILP) problem. They propose a heuristic approach to greedily solve the ILP.
4. Subquery Deltas: based on the assumptions of recurring subqueries, they aggressively cache all

intermediate results of subqueries, and send a diff (delta) between new and old results to reduce data transferring workload.

**Query Planning**

To output a query execution plan, i.e. join order and execution mechanism (broadcast join, semijoin, etc.), in a geo-distributed setting, they extend Apache Calcite to add awareness of geo-distributed execution. Apache Calcite is a centralized SQL query planner, which is being used by several projects including Hive, which their system is built on top of. Calcite takes as input a SQL query parse tree along with basic statistics of each table, and produces a modified, optimized parse tree. Statistics used by Calcite are, for example, the number of rows in each table, the average row size in each table, and an approximate count of the number of distinct values in each column of each table. Calcite uses these statistics along with some uniformity assumptions to optimize join order. Calcite++ extends Calcite for awareness of geo-distributed execution. With additional input like the number of partitions of each table, new rules to compare the cost of various (distributed) join algorithms are introduced in Calcite++. Its output is an optimized join order annotated with the execution strategy for each join that leads to the lowest cost in the geo-distributed setting. The output plan can usually be translated into a directed acyclic graph (DAG), which shows the dependency relationship among data partitions and tasks.

**Pseudo-Distributed Measurement**

The previous phase of query planning will use crude table statistics. However, to make site selection and data replication decisions, much better accuracy in statistics like data transfer volume along edges of DAG is required. To this end, they measure data transfer volumes when executing the plan in the currently deployed configuration. This configuration could be a centralized deployment where all data partitions are copied into a central data center, or a distributed Geode deployment. They modify the query execution when necessary to make it possible to collect statistics, especially in the centralized deployment. Specifically, to make measurements in the centralized deployment, they need to simulate a virtually distributed topology, in which each data partition is in a separate data center. This can be accomplished by rewriting the queries to push down "WHERE country=X" clauses if the partitions are based on different countries. With this technique, which they call "pseudo-distributed measurement", they can inspect intermediate data sizes and identify the data transfer volume along each edge of the DAG corresponding to a query execution plan.

**Site Selection and Data Replication**

Using the above pseudo-distributed measurement technique, they could collect data transfer volume statistics, which are then used for site selection and data replication. Site selection is to specify which data centers tasks should be assigned to for running. Data Replication is to specify which data centers each data partition should be replicated to for performance or fault tolerance. They formulate an integer linear programming problem to jointly solve these two decision making problems with the goal of minimizing the total cost of data transfer. Solving such ILP can be slow. Therefore, they proposed an alternative greedy algorithm that can scale to solve much larger problems.

**Subquery Deltas**

They leverage the storage and computation in each data center to aggressively cache intermediate results of subqueries. For example, if data center A receives the same subquery from data center B again, A will recompute the subquery again, and a diff (delta) between the new and old results. The diff will be sent from A to B instead of the whole result. The purpose of caching is to reduce the data transfer between data centers. It will not reduce the computation within each data center.

## Limitations and Possible Improvements

- They make no attempt to optimize analytics query latency.
- Their solution is only for SQL queries, without consideration for Machine learning applications, which are ubiquitous in today’s data analytics.
- Their solution deals with relational data only.
- They only provide eventual consistency.
- The topology of the WAN is not considered in the paper.

## Summary of Class Discussion

The first concern in our discussion about this paper lies on whether or not caching intermediate results leads to better performance. The answer is -- not always. There are occasions when sending the delta itself is as costly as executing the whole query, for example, when the data is experiencing large updates. Speaking of the geo-distributed setting, lots of issues that are faced in this setting were also faced before in the case of a single data center. For example, when map-reduced came out, the bandwidth within a data center is also limited, like 10-100 MB/s. People proposed systems that would avoid large data transfer among machines within a data center. But now this is no longer a bottleneck within a data center because its bandwidth could reach 10-100 GB/s. Nevertheless, there are some new challenges in the geo-distributed setting like the privacy issue. Another concern in our discussion is whether nor not reducing the amount of data transferred leads to reducing query latency. By reducing data transfer, the query latency is very likely to be reduced as well. However, there are more factors to consider is we want to reduce query latency, such as WAN topology. WAN topology is not a full mesh. Therefore, we can make query execution plan based on the WAN topology so that the utility of the links are increased. We can also replan the requery upon a WAN topology change. This paper didn't consider optimization over multiple queries and the possible contention among them. This point is addressed by the companion paper.