WANalytics: Analytics for a geographically distributed data-intensive world

Ashish Vulimiri*, Carlo Curino+, Brighten Godfrey*, Konstantinos Karanasos+, George Varghese+

* UIUC + Microsoft
Large organizations today: Massive data volumes

• Data collected across several data centers for low end-user latency

• Use cases:
  – User activity logs
  – Telemetry
  – …
Current scales: 10s-100s TB/day

across up to 10s of data centers

<table>
<thead>
<tr>
<th>Company</th>
<th>Data Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>n * 10s TB/day</td>
</tr>
<tr>
<td>Twitter</td>
<td>100 TB/day</td>
</tr>
<tr>
<td>Facebook</td>
<td>15 TB/day</td>
</tr>
<tr>
<td>Yahoo</td>
<td>10 TB/day</td>
</tr>
<tr>
<td>LinkedIn</td>
<td>10 TB/day</td>
</tr>
</tbody>
</table>
Data must be analyzed as a whole

• Need to analyze **all** this data to extract insight

• Production workloads today:
  – Mix of SQL, MapReduce, machine learning, …
Analytics on geo-distributed data: Centralized approach inadequate

Current solution: copy all data to central DC, run analytics there

1. Consumes a lot of bandwidth
   - Cross-DC bandwidth is expensive, very scarce
   - “Total Internet capacity” only \(\approx 100 \text{ Tbps}\)

2. Incompatible with sovereignty
   - Many countries considering making copying citizens’ data outside illegal
   - Speculation: *derived* info will still be OK
Alternative: Geo-distributed analytics

we build system supporting *geo-distributed* analytics execution

- Leave data partitioned across DCs
- Push compute down (distribute workflow execution)
Geo-distributed analytics

Centralized execution: 10 TB/day

Distributed execution: 0.03 TB/day

- t = 0 push down preprocess
- t = 1 distributed semi-join
- t = 2 centralized k-means
Geo-distributed analytics

Centralized execution: 10 TB/day

Distributed execution: 0.03 TB/day

- $t = 0$: push down preprocess
- $t = 1$: distributed semi-join
- $t = 2$: centralized k-means
Geo-distributed analytics

Centralized execution: 10 TB/day

Distributed execution: 0.03 TB/day

333x cost reduction
Building a system for Geo-distributed analytics

• Possible challenges to address:
  – **Bandwidth**
  – Fault tolerance
  – Sovereignty
  – Latency
  – Consistency

• Starting point: system we build targets the batch applications considered earlier
PROBLEM DEFINITION
Computational model

- DAGs of arbitrary tasks over geo-distributed data
- Tasks can be **white box** or **black box**

![Diagram](image-url)
Unique characteristics
(what make this problem novel)

1. Arbitrary DAG of computational tasks

2. No control over data partitioning
   – Partitioning dictated by external factors, e.g. end-user latency

3. Cross-DC bandwidth is only scarce resource
   – CPU, storage within DCs is relatively cheap

4. Unusual constraints:
   – heterogeneous bandwidth cost/availability
   – sovereignty

5. Bulk of load is stable, recurring workload
   – Consistent with production logs
Problem statement

• Support arbitrary DAG workflows on geo-distributed data
  – Minimize bandwidth cost
  – Handle fault-tolerance, sovereignty

• Configure system to optimize given ~stable recurring workload (set of DAGs)
KEY TAKE-AWAY 1:

Geo-distributed analytics is a fun and industrially relevant new instance of classic DB problems
OUR APPROACH
Data transfer optimization: Trading CPU/storage for bandwidth

• Runtime optimization that works irresp of computation

• CPU, storage within DCs is cheap
• Bandwidth crossing DCs is expensive
• This is one way we trade CPU/storage for bandwidth reduction
Data transfer optimization: Caching

• We use aggressive caching: Cache all intermediate output

• If computation recurs:
  – recompute results
  – send \( \text{diff(new results, old results)} \)

• Actually worsens CPU, storage use

• But saves cross-DC bandwidth
  – all we care about
Data transfer optimization: Caching

- Caching naturally helps if one DAG arrives repeatedly (intra-DAG)
- But interestingly: also helps inter-DAG
  - When multiple DAGs share common sub-operations
  - (Because we cache all intermediate output)
- E.g. TPC-CH
  - 5.99x for a part of the workload
Data transfer optimization: Caching ≈ View maintenance

- Caching is a low-level, mechanical form of (materialized) view maintenance

+ Works for arbitrary computation
- Compared to relational view maintenance
  - Is less efficient (CPU, storage)
  - Misses some opportunities
The extreme ratio of bandwidth to CPU/storage allows for novel optimizations.
WORKLOAD OPTIMIZER
Robust evolutionary approach

- Start by supporting existing “centralized” plan
- Continuous adaptation (loop):
  - Come up with a set of alternative hypotheses
  - Measure their costs using pseudo-distributed execution
    - Novel mechanism with zero bandwidth-cost overhead
  - Compute new best plan
    - Execution strategy
    - Data replication strategy
  - Deploy new best plan
Robust evolutionary approach

• Start by supporting existing “centralized” plan
• Continuous adaptation (loop):
  – Come up with a set of alternative hypotheses
  – Measure their costs using *pseudo-distributed execution*
    • Novel mechanism with zero bandwidth-cost overhead
  – Compute new best plan
    • Execution strategy
    • Data replication strategy
  – Deploy new best plan
Robust evolutionary approach

• Start by supporting existing “centralized” plan

• Continuous adaptation (loop):
  – Come up with a set of alternative hypotheses
  – Measure their costs using *pseudo-distributed execution*
    • Novel mechanism with zero bandwidth-cost overhead
  – Compute new best plan
    • Execution strategy
    • Data replication strategy
today
  – Deploy new best plan

(for rest see paper)
Optimizing execution: Subproblem definition

• Given:
  – Core workload: a set of recurrent DAGs
  – Sovereignty, fault-tolerance requirements

• Need to decide best choice of:
  – Strategy for each task (e.g. hash join vs semi join)
  – Which task goes to which DC
Optimizing execution: Difficulties

1. Optimizing even one task in isolation is very hard

2. Should jointly optimize all tasks in each DAG

3. Should jointly optimize all DAGs in workload
   - Caching

4. Sovereignty, fault-tolerance
Optimizing execution:

Difficulties

1. Optimizing even one task in isolation is very hard

DAG:

P

Q

Data:

DC

P

Q

Optimal distributed join algo for P $\bowtie$ Q

Q update rate (GB/OLAP run)

P update rate (GB/OLAP run)

replicate P

distributed hash join

centralize

replicate Q
Optimizing execution: Difficulties

1. Optimizing even one task in isolation is very hard

2. Should jointly optimize all tasks in each DAG

3. Should jointly optimize all DAGs in workload
   - Recall: caching helps when DAGs share sub-operations

4. Sovereignty, fault-tolerance
Optimizing execution: Greedy heuristic

• Process all DAGs in parallel, separately. In each DAG:
  – Go over tasks in topological order
  – For each task, greedily pick lowest-cost available strategy
When does the greedy heuristic work?

- **Contractive DAGs**: picks optimal strategy
  - make up 98% of DAGs in our experiments
When does the greedy heuristic work?

- Contractive DAGs: picks optimal strategy [98%]
- DAGs that expand then contract: may not [2%]
Optimizing execution: Beyond the heuristic

• Have a precise ILP formulation for special cases
  – SQL-only DAGs
  – MapReduce-only DAGs
  – (Handles fault-tolerance and sovereignty as constraints)

• Alternate heuristics

• General problem remains open
KEY TAKE-AWAY 3:

The optimization space is massive, yet simple heuristics seem to yield good results.
EVALUATION
Prototype: WANalytics

• Implemented Hadoop-stack prototype
  – MapReduce, Hive, OpenNLP, Mahout, …

• Experiments up to 10s of TBs scale
  – Real Microsoft production workload
  – Three standard synthetic benchmarks:
    BigBench, TPC-CH, Berkeley Big-Data
  – Mix of relational and non-relational
Data transfer

TB (raw, uncompressed)

Size of OLTP updates since last OLAP run

Results: BigBench

Centralized

Distributed: no caching

Distributed: with caching

330x
Results: TPC-CH

Data transfer
TB (compressed)

Size of OLTP updates since last OLAP run

Centralized
Distributed: no caching
Distributed: with caching

360x
Results: Microsoft production workload

- Centralized
- Distributed: no caching
- Distributed: with caching

Size of OLTP updates since last OLAP run

Data transfer

257x
Results: Berkeley Big-Data

- Centralized
- Distributed: no caching
- Distributed: with caching

Data transfer
TB (compressed)

Size of OLTP updates since last OLAP run

TB (raw, uncompressed)

3.5x
KEY TAKE-AWAY 4:

The opportunity here is substantial: more than two orders of magnitude in many workloads.
OPEN PROBLEMS
Open Problems

• Evolve optimizer beyond greedy
• Even more general computational models
  – e.g. iteration

• Latency
• Consistency
• Sovereignty / privacy
Open Problems

• Evolve optimizer beyond greedy
• Even more general computational models
  – e.g. iteration

• Latency
• Consistency
• Sovereignty / privacy
Sovereignty: Partial support

• Our system respects “data-at-rest” regulations (e.g., German data should not be stored outside of Germany)

• But we allow arbitrary queries on the data

• Limitation: we don’t differentiate between
  – Acceptable queries, e.g. “what’s the total revenue from each city”
  – Problematic queries, e.g. SELECT * FROM Germany
Sovereignty: Partial support

- Solution: either
  - Legally vet the core workload of queries/views
  - Use differential privacy mechanism

- Open problem
KEY TAKE-AWAY 5:

This is just the first step, lots of related work, lots of fun work ahead
Related Work

- Distributed and parallel databases
- Single-DC frameworks (Hadoop/Spark/…)
- Data warehouses
- Scientific workflow systems
- Sensor networks
- Stream-processing systems
- …
Unique characteristics
(what make this problem novel)

1. Arbitrary DAG of computational tasks

2. No control over data partitioning
   – Partitioning dictated by external factors, e.g. end-user latency

3. Cross-DC bandwidth is only scarce resource
   – CPU, storage within DCs is relatively cheap

4. Unusual constraints:
   – heterogeneous bandwidth cost/availability
   – sovereignty

5. Bulk of load is stable, recurring workload
   – Consistent with production logs
Summary

• Centralized analytics is becoming untenable
• Proposal: geo-distributed analytics execution
• WANalytics, our system, introduces
  – Pseudo-distributed measurement
  – Joint multi-query + redundancy optimization
  – Caching
• On real and synthetic workloads:
  up to 360x less bandwidth than centralized
• Many challenges remain