CS 6453: Geode and Clarinet

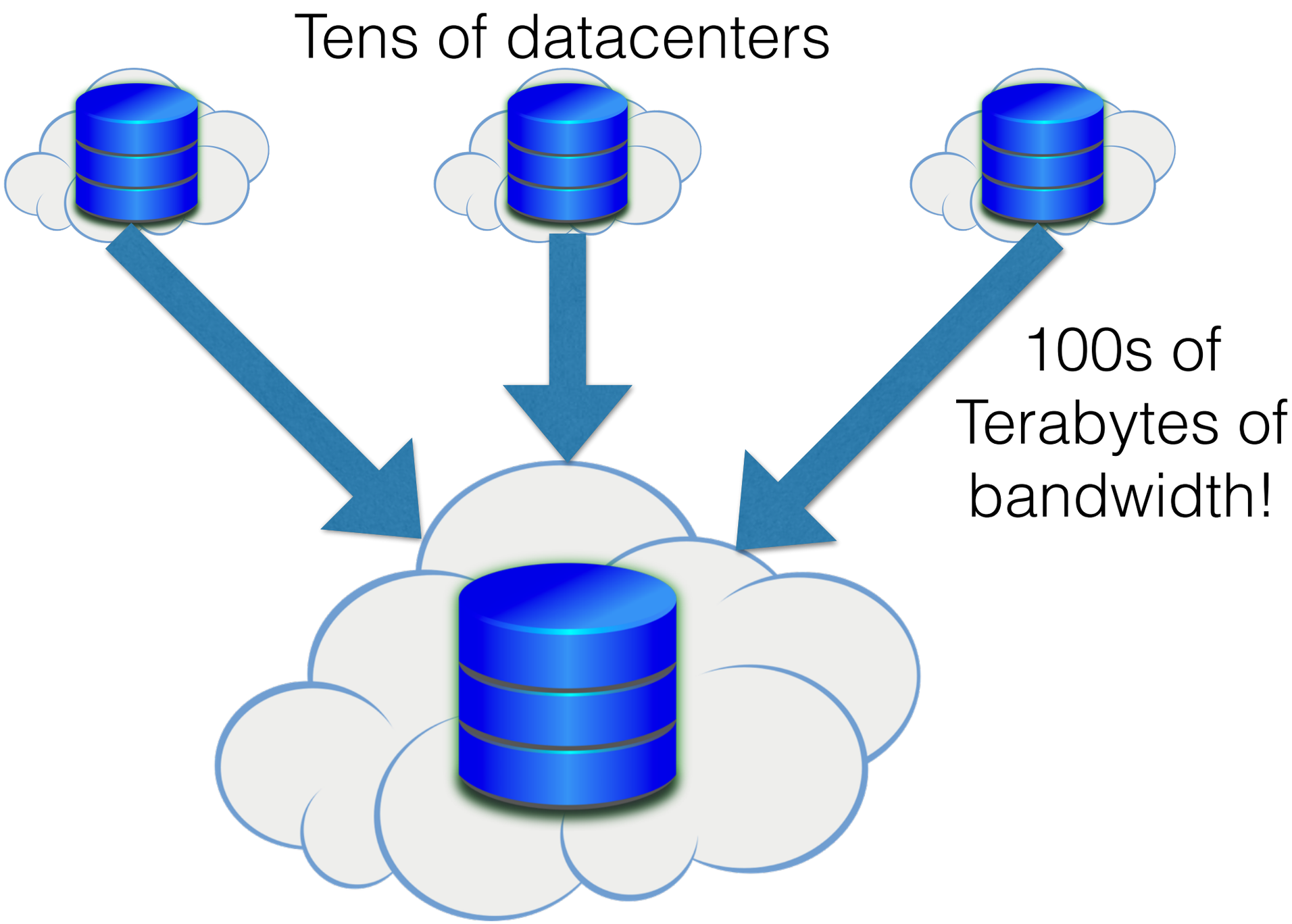
Soumya Basu
April 13, 2017

Motivation

Motivation


NORTH PACIFIC OCEAN
NORTH ATLANTIC OCEAN
SOUTH ATLANTIC OCEAN
INDIAN OCEAN
SOUTH PACIFIC OCEAN
SOUTHERN OCEAN
ARCTIC OCEAN
CANADA
UNITED STATES
GREENLAND (Kalaallit Nunaat) (Denmark)
RUSSIAN FEDERATION
THE WORLD
MERCATOR PROJECTION
scale at the Equator

Status Quo



Why is this a problem?

- Application demands are growing
- Wide Area Network capacity is growing *more slowly* than Datacenter bisection bandwidth
  - (2015)1 Pb/s for datacenters vs 100 Tb/s for WAN
- Different jurisdictions are getting more protective about data
  - Might be *illegal* to use this approach for analytics
  - Assumption: Derived data is OK to share

Geode

Related Work

- Lots of prior work on distributed databases
  - Always assumed that databases were in a LAN
  - Transactional workloads (arbitrary, random queries)
    - Geode assumes that queries change slowly

Related Work

- All prior work lacks some key feature that Geode provides
- Solutions that don't focus on bandwidth costs
  - Spanner, Mesa, RACS
- Solutions that don't handle the relational database model
  - Jetstream, Volley
- Solutions that don't handle multi-cloud scenarios
  - Hive, Pig, Spark

Batch Analytics Requirements

- Optimize bandwidth costs
- Constraints:
  - Sovereignty: Laws preventing data migration
  - Fault-tolerance: May have some replication
- Non-issues: latency, consistency

More Assumptions

- Data Birth: Cannot intelligently partition the data-locations are given
- Fixed Queries, but supports slowly changing query workload
  - e.g. finding the top 10 bestselling books every day
- Inter-Datacenter Bandwidth is scarce
  - Intra-datacenter bandwidth, cpu, storage free

Contributions

- Subquery deltas
- Pseudo-distributed measurement
- Query optimization

Subquery Deltas

- Cache all subqueries sent across datacenters
  - Subsequent queries are recomputed at the origin
  - Origin only sends the *diff* between the old and new output
- In TPC-H, this saves 3.5x bandwidth on 6 of the queries

Pseudo-distributed measurement

- How much data will be sent across the WAN for a particular query?
  - If queries stay the same, can create a plan per query
- Two insights to make this measurement possible
  - Insert a WHERE clause into each SQL query to simulate per-partition output
  - Ignore partial aggregation in datacenters

Query Optimization

- Centralized query planning from distributed database literature
  - Change cost functions based on bandwidth measurements
- Two other problems
  - Site Selection: Where to run each task
  - Data Replication: Where copies are stored

Query Optimization (cont)

- Naive approach: solve both problems using ILP
  - Solver timeout of 1 hour only handles ~10 datacenters
- Greedy heuristic for site selection: pick the site where copying over the input data is cheapest
- Use simple ILP to solve data replication

Limitations

- Weak consistency is not useful for many types of applications
- Completely ignores underlying privacy reasons behind data migration
- Many step query analytics not expressible in Geode
  - This is solved by our next paper!

Clarinet

Problem Statment

- Same geo-distributed setting as Geode
- Clarinet minimizes query response time
  - Where a query takes ~seconds-minutes to run
- WAN bandwidth is taken into account in model
- Supports richer analytics queries than Geode (multi-stage queries)

Technical Contributions

- Main insight: Let database incorporate WAN into evaluation of query plans
- Three techniques introduced:
  - Late binding of the evaluation plan
  - Task Scheduling
  - Handling resource fragmentation

Late Binding

- Normal query optimizer steps:
  - Generate possible query plans
  - Score all plans and pick the best one
  - Map the logical plan to a physical plan and execute

Late Binding

- Clarinet query optimizer steps:
  - Generate possible query plans
  - ~~**Score all plans and pick the best one**~~
  - Map *all* logical plans to physical plans
  - Score all *physical* plans and pick the best one

Multi-Query Late Binding

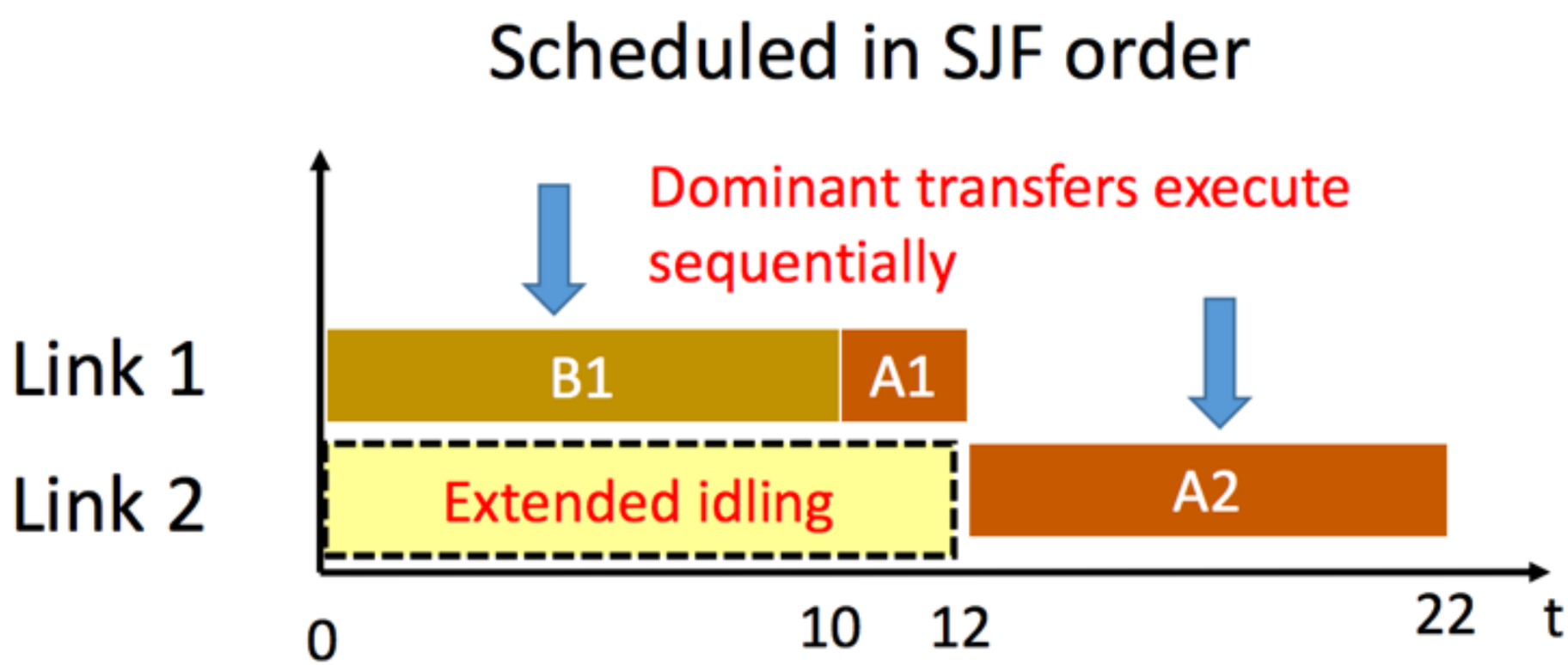
- Generate possible query plans
- Map *all* logical plans to physical plans, for all queries
- Score all *physical* query plans, pick the shortest one
- Reserve bandwidth on the network for that query
- Repeat full process to pick the next query

Task Placement

- Decided one stage at a time, minimizing per stage runtime
- Scheduling of network transfers done by solving an ILP
  - Allows Clarinet to encode transfer dependencies
- Doing task placement across queries is handled the same way

Resource Fragmentation

- Naive network schedule simply follows the order the network was reserved in Late Binding step
  - This is Shortest Job First



Resource Fragmentation

- Relaxation of SJF to k-SJF
- Keep track of the k shortest jobs
- If any of those flows are able to be scheduled, start it immediately
  - Fairness issue for long jobs, so add a deadline based heuristic to make things better
- k has a sweet-spot to not increase average job completion time

Limitations

- WAN Bandwidth varies, so assuming its constant is a bad assumption
- Resource fragmentation solution is very ad-hoc
- Not sure what the absolute numbers are in evaluation
  - Query response times decrease by 50%

Holy Grail

- Interactive transactions
  - Both papers use ILP somewhere, so this technique would not work
  - Most of the overheads would be very stark with respect to the query processing time