Optimizing Data Placement for Distributed Computation

UNIVERSITY OF WATERLOO

uwaterloo.ca

Lukasz Golab

Based on

- Lukasz Golab, Marios Hadjieleftheriou (AT&T), Howard Karloff (Yahoo), Barna Saha (AT&T), *Distributed data placement to minimize communication costs via graph partitioning*, SSDBM 2014
- Available at www.engineering.uwaterloo.ca/~lgolab

Background

- Popular big data trend
  - Shared-nothing clusters of servers
  - Distributed storage and processing
  - Great for jobs that parallelize easily
    - E.g., count the number of documents containing some string
  - But data-intensive jobs require data migration

Background

- CoHadoop [VLDB 2011, El-Tabakh et. al.]
  - User can give file co-location hints
- Our Goal
  - Given the query workload, can we automatically place the data on a computing cluster to minimize data transfer cost?

Problem Statement

- m queries, Q_1 through Q_m
- n tables, T_1 through T_n,
  - then ith table having size w_i
- A query requires one or more tables
- For each Q_i and T_j required by Q_i, a data transfer volume C_ij

Problem Statement

- In general,
  - Table = data item, file, table partition, etc.
  - Query = anything that processes data, etc.

Problem Statement

- k servers, S_1 through S_k,
  - the jth server having storage capacity s_j and processing capacity p_j
- Every query runs on a server
  - copies the data it needs to its server
  - does some processing
- Every table is stored on a server
  - we'll get to replication later

Problem Statement

- Assign each query and table to a server in a way that
  - minimizes the the total data transfer cost during query execution
  - and does not violate the server storage and processing capacities

Assumptions

- Storage capacity of any one server < sum(w_i), the total size of the tables
- Processing capacity of any one server < m (the number of queries)
- Otherwise, just use one server, and data transfer cost = 0
- Queries are data-intensive

UNIVERSITY OF WATERLOO

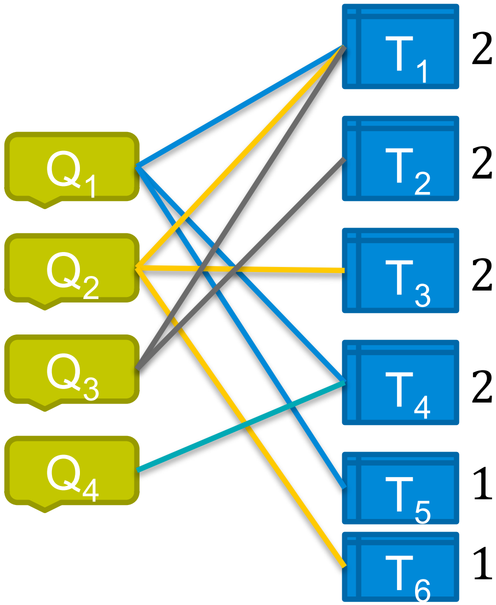
Solution #1

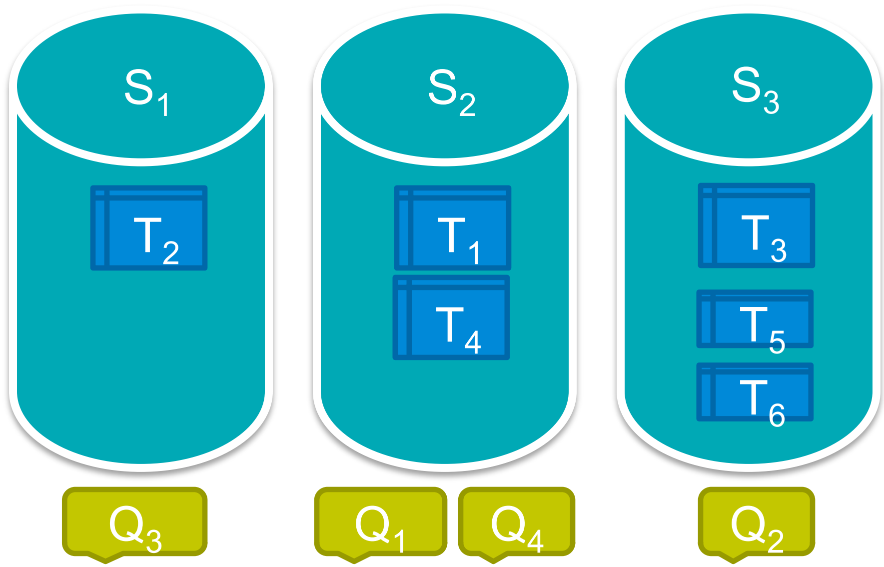
- Formulate an optimization problem and solve it using CPLEX
  - Extremely slow due to complexity of the problem (NP-hard, as we prove in the paper)

Solution #2

- Compute an approximate solution
- Reduce our problem to graph partitioning
  - Still NP-hard but efficient approximation algorithms exist
  - E.g., METIS

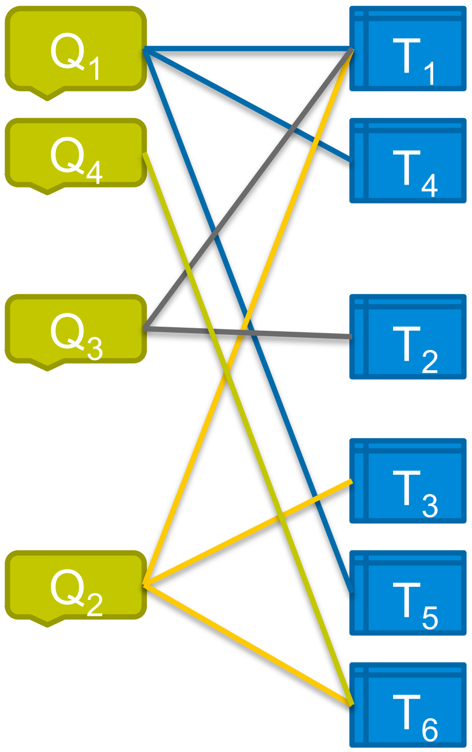
Example





UNIVERSITY OF WATERLOO

Bipartite Graph Partitioning



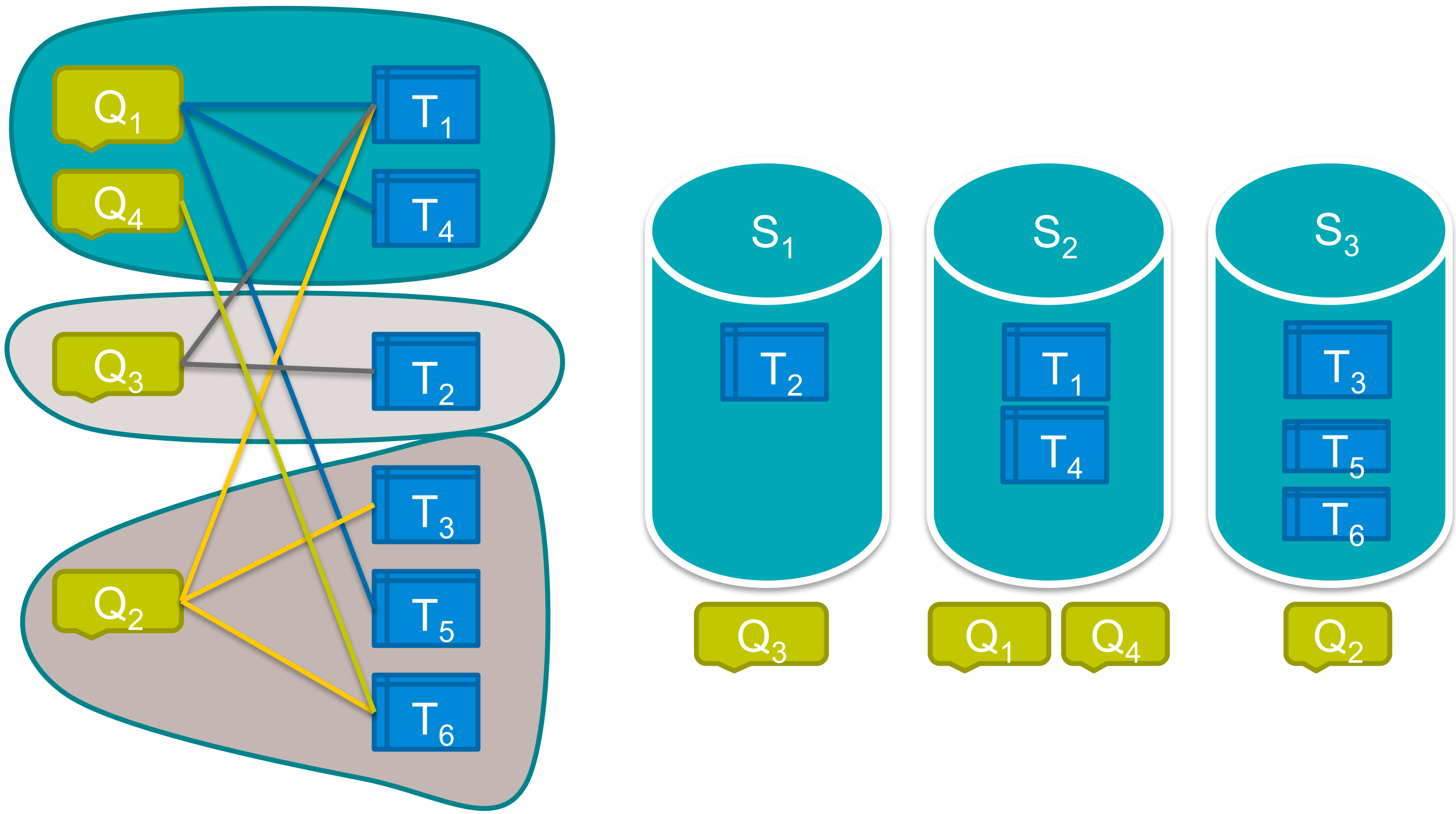
- Queries on the left
- Tables on the right
- Each query node has processing weight 1
- Each table node has storage weight w_i
- Each edge from Q_i to T_j has weight C_ij

UNIVERSITY OF WATERLOO

Reduction to Graph Partitioning

- Partition the query graph:
  - Into k parts
  - Each with sum(w_i) <= server storage capacity and num. queries <= server processing capacity
  - To minimize the weight of the cut edges
- Claim: this reduction solves our problem

Example

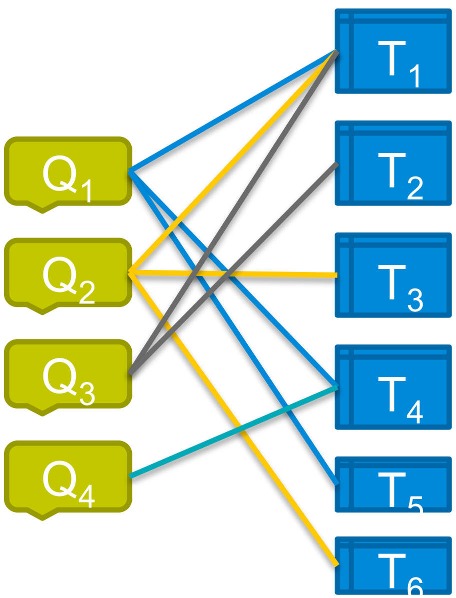


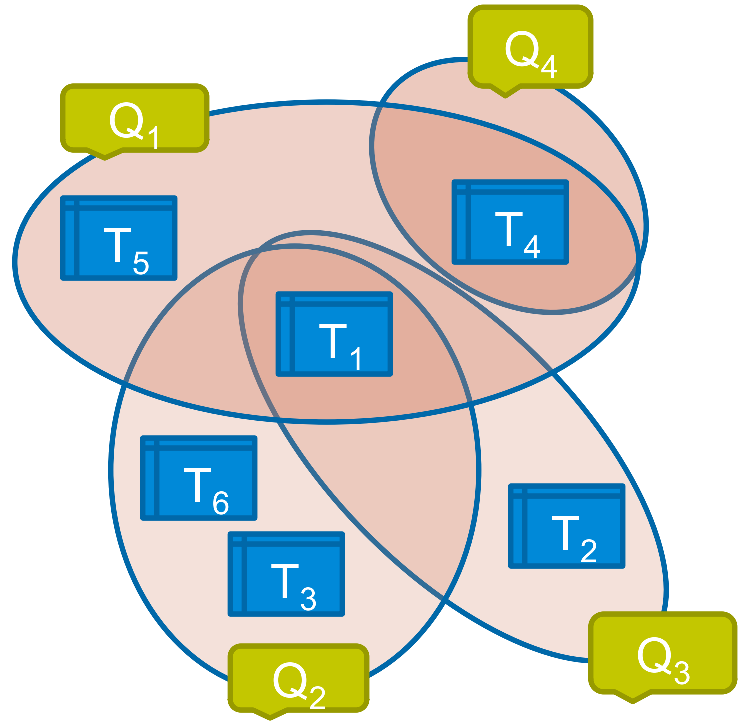
Previous Work

- OLTP setting: minimize the number of distributed transactions [VLDB 2010, Curino et. al.]
- Modeled as hypergraph partitioning
  - More general than graph partitioning → worse performance

Hypergraph Partitioning

- Tables are nodes
- Queries are hyperedges
- Cost of cutting a hyperedge = 1





UNIVERSITY OF WATERLOO

Replication

- What if we store up to or exactly r copies of each table?
- Optimization program gets even more complex and slow
- We propose 2 algorithms using graph partitioning as a subroutine

Algorithm #1

- Pretend the server capacities are s_i/r and p_i/r
- Run *graph partitioning* once
  - – Place one copy of each table
- Randomly permute the servers
  - – Place second copy of each table
- …

Problem with Algorithm #1

- Some tables may end up with < r copies
- E.g.,
  - 1-2-3-4-5-6-7-8
  - 1-7-5-6-2-8-4-3
  - 5-3-1-8-6-7-4-2

Algorithm #2

- Partition servers into r groups
- Run *graph partitioning* using the first group of k/r servers
- Remove the m/r cheapest queries
- Run *graph partitioning* using the second group of k/r servers
  - Repeat with ALL tables but only the remaining queries
- Remove the m/r cheapest queries
- …

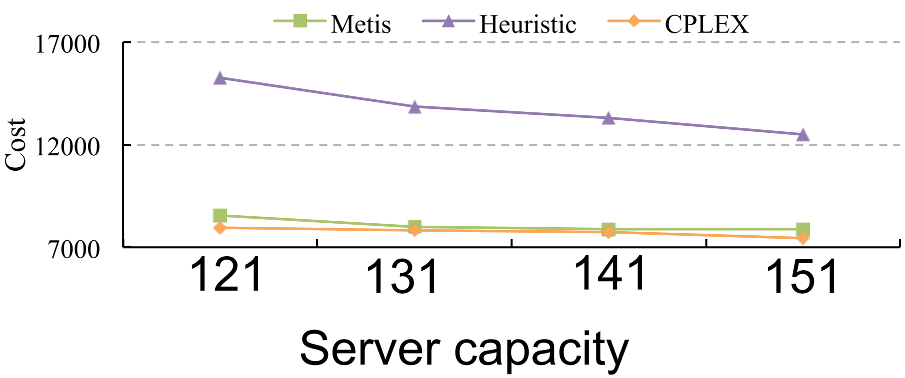
UNIVERSITY OF
WATERLOO

Experimental Results

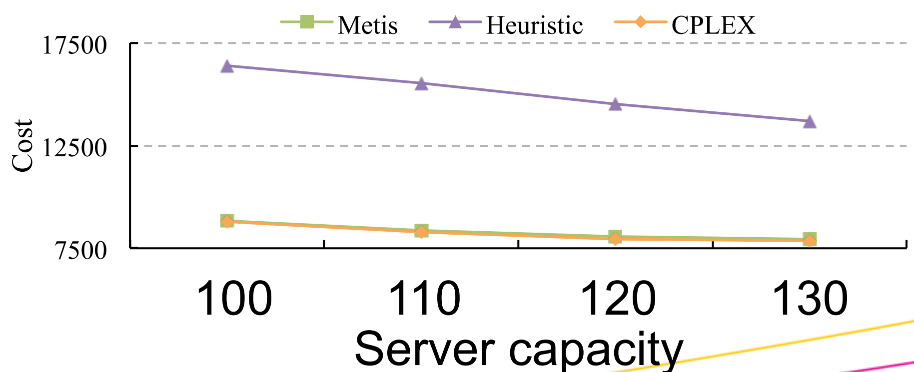
- Optimization program solved by CPLEX vs. graph partitioning solved by METIS vs. simple heuristic
  - Using a workload similar to TPC-DS (24 tables, 99 queries)
- Scalability experiments using very large random query graphs

Sample Results

- 8 servers



16 servers

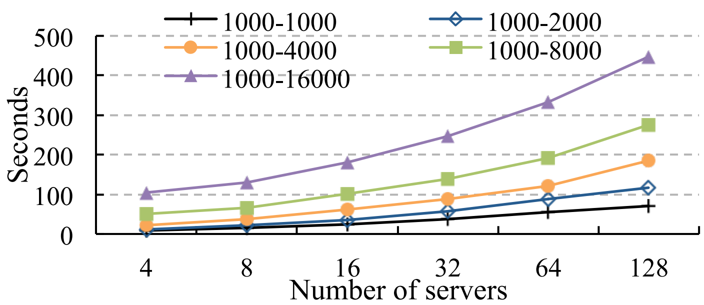


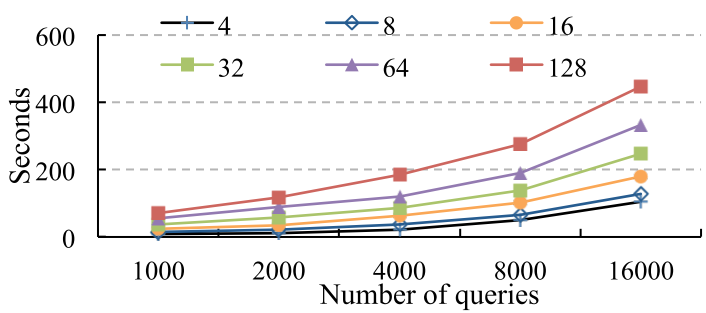
UNIVERSITY OF WATERLOO

Scalability

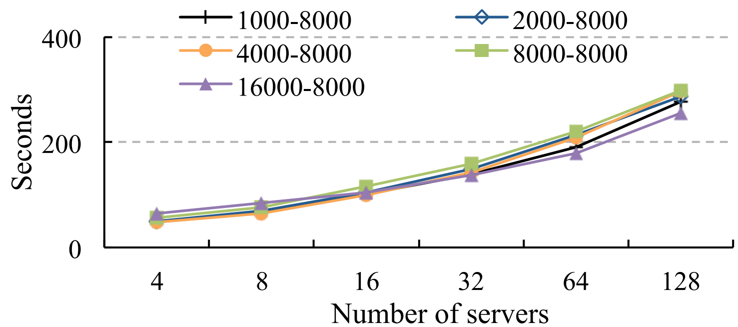
- number of queries: 1000-16000
- number of tables: 1000-16000

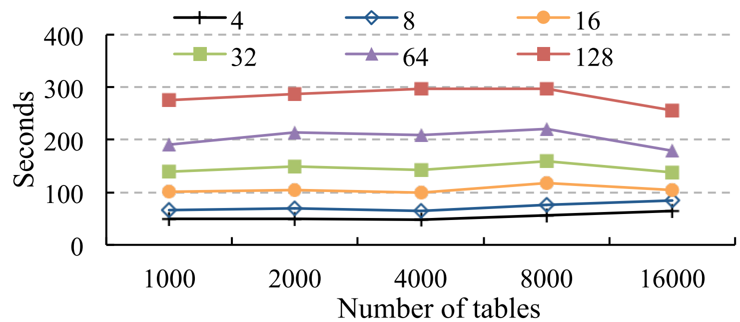
Queries





Tables





UNIVERSITY OF WATERLOO

See Paper For

- Replication algorithm #2 is better
- Extension to complex workflows
  - Intermediate results

Summary

- Careful data placement is necessary when running data-intensive queries on a cluster
- Provided data placement algorithms via graph partitioning
- Future work: combine table/query partitioning with data placement