Data Sharing: Querying and Linking Distributed and Autonomous Data

Yang Cao, Wenfei Fan, Peter Buneman

University of Edinburgh

Background

Distributed databases: healthcare, business intelligence, e-government, ...

- *Tasks*: querying, linking and sharing
- *Data*: distributed, heterogeneous, large scale

Challenges:

- **Privacy & security**: data owners ("private") vs. users ("open")
- **Heterogeneity**: relations, key-value, graphs
- **Scalability**: limited resources vs. big data analytics
- **Accountability**: ownership and accountability of shared data collection

Querying Shared Data with Security Heterogeneity [5]

Challenges

- heterogeneous security requirements
- centralized evaluation not possible

Solution:

1. Modeling data sharing protocols
2. Query evaluation under protocols
   - heterogeneous distributed query plans
   - optimal security-efficiency trade-offs
   - leverage security heterogeneity in plans

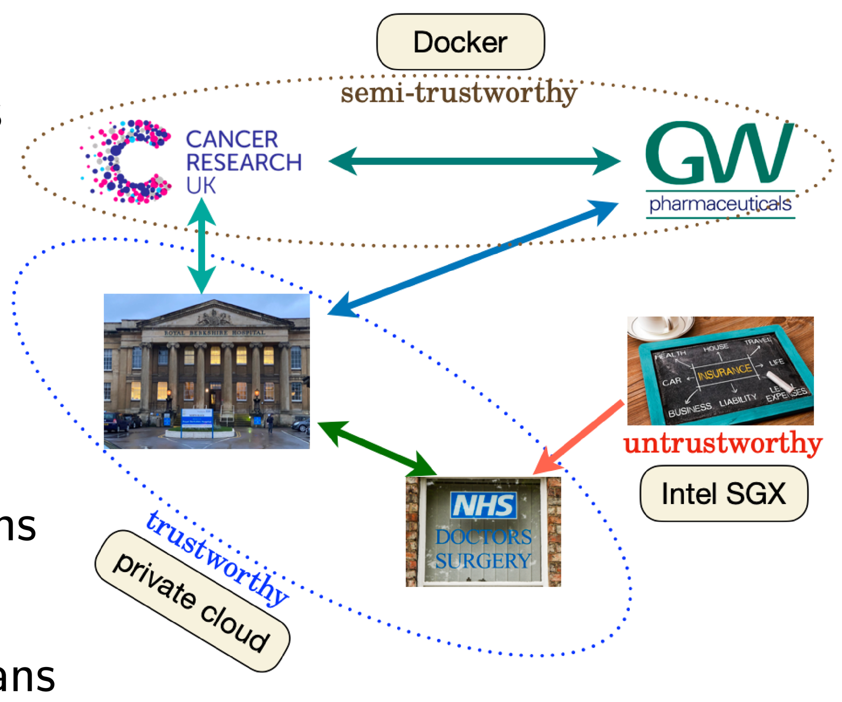

Figure: Heterogeneous data sharing

Data sharing protocols ρ specifying:

- capsules: logic units for computations over shared data
- hosts: data owners that host capsules
- pair-wise privacy requirements:
  - permitted capsule specifications
  - secure communication overheads

Planning under protocols: find optimal plans with bounded security *toll*:

[Complexity]:

1. decidable in NEXPTIME;
2. PSPACE-hard even when ρ is linear;
3. Σ_3^p-hard even when Q is in **SPC** and ρ is linear.
   Moreover, 2 and 3 hold even when $\mathcal{S}$ has two sites only.

Intractable to make the best (optimal) use of heterogeneity in data sharing.

[Algorithm]: Nonetheless, a two phase approach with guarantees:
Step (1): generating toll-minimized distributed plan ξ_Q

- toll-minimized ξ_{op} for each algebra operation of Q
- an $O(\log n)$-approximation algorithm for ⋈

Step (2): optimizing ξ_Q within the toll budget

- via an automic operaton κ for "rebalancing" ξ_Q
- a near-optimal design of κ (2-approximation of the optimal for ⋈)

[Effectiveness]: it speeds up state-of-the-art secure database system (SMCQL) by 18+ times over 1GB of **TPCH** data.

Heterogeneous Entity Resolution [6]

Main task: link entities across a relational database D and a graph G

Challenge: traditional ER methods work for relations only (homogeneous).

HER: decide whether tuple $t \in D$ matches vertex v of G:

1. convert relations D into a graph G_D
2. whether t matches $v \Rightarrow$ whether v_t in G_D matches v in G
3. parametric simulation between G_D and G
   - graph simulation extended with label matching functions
   - remains in quadratic-time
   - learned threshold for score functions
   - parallelizable to scale over large graphs and relations

Scaling via Consistent Caching [2]

Background:

- large communication overhead and limited resources per node
- transactional accesses (workloads) become prevalent

Main task: scale out via data cache with consistency guarantees

Results:

- prove traditional policies (e.g., LRU) are not competitive for transactions
- formulate consistent cache scheme; show it's NP-complete even for uni-size accesses, in contrast to trivially PTIME for conventional caching
- develop a consistent cache policy that is theoretically competitive and optimal when transactions access data items of unit size
- transaction batching and reordering for caching
- implemented and tested with Memcached@HBase: 126% improvement on average for transaction throughput, while guaranteeing consistency

Monotonic Consistent Caching [1]

Overview: further extend [2] to support

- both monotonicity and consistency for transactions;
- three input models: batch, semi-online and online.

Results:

- formulate monotonic consistent cache (MCC) scheme for transactions
- settle its complexity (NP-complete) and develop a characterization
- as an application of the characterization, develop an MCC policy for the batch model that is provably optimal for transactions
- for semi-online (reads offline writes online) and online models, develop MCC policies that take *blackbox binary* ML classifier as an oracle M:
  - [Competitiveness] if M is accurate, the MCC policy is theoretically competitive (for online) and even optimal (for semi-online); and
  - [ML-robustness] if M is adversarial (*i.e.*, each and every prediction M produces is incorrect), the MCC policy is still theoretically competitive for both semi-online and online models.
- implemented for Redis@HBase: improve transaction throughput by 77.15% while guaranteeing consistency and monotonicity

Accountability of Shared Curated Data [4, 3]

Background:

- datasets are often *shared/copied* $\rightarrow$ *modified* $\rightarrow$ *published* $\rightarrow \cdots$
- a data collection contains contributions from multiple users
- contributions form a dependency hierarchy (copy-modify-contribute)
- however, entire dataset is often treated as one single "article"

Main task: how to account the ownership of pieces of data in a dataset

Results:

- a model of citation graph for databases
- method for generating data summaries of "optimal" granularity
  - stress measures of data summaries
  - compute accountability by measuring stress
- application to a collectively curated pharmacology database (GtoPdb)

Publications

[1] S. An and Y. Cao. Making cache monotonic and consistent. In *VLDB*, 2023.

[2] S. An, Y. Cao, and W. Zhao. Competitive consistent caching for transactions. In *ICDE*, 2022.

[3] P. Buneman, D. Dosso, M. Lissandrini, and G. Silvello. Data citation and the citation graph. *Quant. Sci. Stud.*, 2(4), 2021.

[4] P. Buneman, D. Dosso, M. Lissandrini, and G. Silvello. Expanding the citation graph for data citations. In *SEBD*, pages 276–283, 2022.

[5] Y. Cao, W. Fan, Y. Wang, and K. Yi. Querying shared data with security heterogeneity. In *SIGMOD*, 2020.

[6] W. Fan, L. Geng, R. Jin, P. Lu, R. Tugay, and W. Yu. Linking entities across relations and graphs. In *ICDE*, 2022.