Noname manuscript No.
(will be inserted by the editor)

Is Big Data Analytics beyond the Reach of Small Companies?

Yang Cao, Wenfei Fan, Tengfei Yuan

Received: date / Accepted: date

Abstract Big data analytics is often prohibitively costly. It is typically conducted by parallel processing with a cluster of machines, and is considered a privilege of big companies that can afford the resources. This position paper argues that big data analytics is accessible to small companies with constrained resources. As an evidence, we present BEAS, a framework for querying big relations with constrained resources, based on bounded evaluation and data-driven approximation.

1 Introduction

Big data analytics is expensive. Textbooks tell us that a computation problem is *tractable* if there exists a polynomial-time algorithm for it, *i.e.,* its cost can be expressed as a polynomial in the size n of the input [20]. However, it is no longer the case when it comes to big data. When n is big, algorithms in $O(n^2)$ or even $O(n)$ time may take too long to be practical. Indeed, assuming the largest Solid State Drives (SSD) with 12GB/s for read [19], a linear scan of a dataset of 15TB takes more than 20 minutes. It easily takes hours to join tables with millions of tuples [21]. In other words, many computation problems that are typically considered tractable may become infeasible in the context of big data.

One might be tempted to think that parallel computation could solve the problem, by adding more processors when needed. However, small businesses often have constrained resources and cannot afford large-scale parallel computation.

Is big data analytics a privilege of big companies? Is it beyond the reach of small companies that can only afford constrained resources?

We argue that big data analytics is possible under constrained resources. As an example, we consider relational query answering. Given an SQL query Q and a relational database D, it is to compute the answers $Q(D)$ to Q in D. Relational data accounts for the majority of data in industry. Moreover, it is nontrivial to compute $Q(D)$. Indeed, it is NP-complete to decide whether a given tuple t is in $Q(D)$ when Q is a simple SPC query (selection, projection and Cartesian product), and is PSPACE-complete when Q is in relational algebra, both subsumed by SQL.

Yang Cao[1], Wenfei Fan[1,2], Tengfei Yuan[1]
[1]University of Edinburgh, [2]Beihang University
E-mail: {yang.cao@, wenfei@inf., tengfei.yuan@}ed.ac.uk

We propose BEAS, a new query evaluation paradigm to answer SQL queries under constrained resources. The idea is to make big data small, *i.e.,* to reduce queries on big data to computation on small data. Underlying BEAS are two principled approaches: (1) bounded evaluation that computes exact answers by accessing a bounded amount of data when possible [3,5,8,11,12], and (2) a data-driven approximation scheme that answers queries for which exact answers are beyond reach under bounded resources, and offers a deterministic accuracy bound [4].

As proof of concept, we have developed a prototype system [7] and evaluated it with call-detailed-record (CDR) queries at Huawei. We find that bounded evaluation improves the performance of CDR queries by orders of magnitude [22], and is able to reduce big datasets from PB (10^{15}) to GB (10^9) in many cases.

Below we give a brief introduction to bounded evaluation (Section 2), followed by data-driven approximation scheme (Section 3). Putting these together, we present BEAS, our resource-bounded query evaluation framework (Section 4).

2 Bounded Evaluation

Given a query Q posed on a big dataset D, bounded evaluation [11] aims to compute $Q(D)$ by accessing only a bounded subset D_Q of D that includes necessary information for answering Q in D, instead of the entire D. To identify D_Q, it makes use of an access schema $\mathcal{A}$, which is a set of access constraints, *i.e.,* a combination of simple cardinality constraints and their associated indices.

Under access schema $\mathcal{A}$, query Q is *boundedly evaluable* if for all datasets D that conform to $\mathcal{A}$, there exists a fraction $D_Q \subseteq D$ such that

- $Q(D_Q) = Q(D)$, *i.e.,* D_Q suffices for computing *exact answers* $Q(D)$; and
- the time for identifying D_Q and hence the size $|D_Q|$ of D_Q are determined by Q and $\mathcal{A}$ only. That is, the cost of computing $Q(D_Q)$ is independent of $|D|$.

Intuitively, $Q(D)$ can be computed by accessing D_Q. We identify D_Q by reasoning about the cardinality constraints in $\mathcal{A}$, and fetch it by using the indices in $\mathcal{A}$.

Example 1: Consider a database schema $\mathcal{R}_0$ consisting of three relations:

(a) person(pid, city), stating that pid lives in city,
(b) friend(pid, fid), saying that fid is a friend of pid, and
(c) poi(address, type, city, price), for the type, price and city of points of interest.

An example access schema $\mathcal{A}$ consists of the following two access constraints:

- ψ_1: friend(pid $\to$ fid, 5000),
- ψ_2: person(pid $\to$ city, 1).

Here ψ_1 is a constraint imposed by Facebook [18]: a limit of 5000 friends per person; and ψ_2 states that each person lives in at most one city. An index is built for ψ_1 such that given a pid, it returns all fid*s* of pid from friend, *i.e.,* ψ_1 includes the cardinality constraint (5000 fid*s* for each pid) and the index; similarly for ψ_2.

Consider a query Q_1 to *find the cities where my friends live*, which is taken from Graph Search of Facebook [10]. Written in SQL, Q_1 can be expressed as:

```
select  p.city
from    friend as f,  person as p
where   f.pid = p0 and f.fid = p.pid
```

where p_0 indicates "me". When an instance D_0 of $\mathcal{R}_0$ is "big", *e.g.,* Facebook has billions of users and trillions of friend links [18], it is costly to compute $Q_1(D_0)$.

However, we can do better since Q_1 is boundedly evaluable under $\mathcal{A}$: (a) we first identify and fetch at most 5000 $\mathsf{fid}s$ for person p_0 from relation friend by using the index for ψ_1, and (b) for each fid fetched, we get her city by fetching 1 tuple from relation person via the index for ψ_2. In total we fetch a set D_Q of 10,000 tuples, instead of trillions; it suffices to compute $Q_1(D_0)$ by using D_Q only [3]. $\Box$

The notion of access schema was proposed in [12], and the foundation of bounded evaluation was established in [5, 11]. The theory of bounded evaluation was first evaluated using SPC queries [8] and then extended to relational algebra (RA) [3]. A challenge is that it is undecidable to decide whether an SQL query is boundedly evaluable under an access schema $\mathcal{A}$ [11]. To cope with this, an *effective syntax* $\mathcal{L}$ was developed for boundedly evaluable RA queries [3]. That is, $\mathcal{L}$ is a class of RA queries such that under $\mathcal{A}$,

(a) an RA query Q is boundedly evaluable if and only if it is equivalent to a query Q' in $\mathcal{L}$; and
(b) it takes PTIME (polynomial time) in the size $|Q|$ of Q and size $|\mathcal{A}|$ of $\mathcal{A}$ to check whether Q is in $\mathcal{L}$, reducing the problem to syntactic checking.

That is, $\mathcal{L}$ identifies the *core* subclass of boundedly evaluable RA queries, without sacrificing their expressive power. This is analogous to the study of safe relational calculus queries, which is also undecidable [1]. Based on the effective syntax, we can efficiently check whether a query Q is boundedly evaluable, and if so, generate a query plan to answer Q by accessing a bounded fraction D_Q of D [3].

It has been shown that bounded evaluation can be readily built on top of commercial DBMS (database management systems) such as MySQL and PostgreSQL. It extends the DBMS with an immediate capability of querying big relations under constrained resources [3]. A prototype system was developed in [7]. We find that about 77% of SPC queries [8] and 67% of SQL queries [3] are boundedly evaluable. Better yet, more than 90% of the CDR queries at Huawei are boundedly evaluable.

3 Data Driven Approximation

For queries Q that are not boundedly evaluable, can we evaluate Q against a big dataset D under constrained resources? We answer the question in the affirmative.

We propose a data-driven scheme for approximate query answering. It is parameterized with a *resource ratio* $\alpha \in (0, 1]$, indicating that our available resources can only access an α-fraction of big D. Given α, D and a query Q over D, it identifies $D_Q \subseteq D$, and computes $Q(D_Q)$ and a ratio $\eta \in (0, 1]$ such that

(1) $|D_Q| \leq \alpha|D|$, where $|D_Q|$ is measured in its number of tuples; and
(2) $\mathsf{accuracy}(Q(D_Q), Q, D) \geq \eta$.

Intuitively, it computes approximate answers $Q(D_Q)$ by accessing at most $\alpha|D|$ tuples in the entire process. Thus it can scale with D when D grows big by setting α small. Moreover, $Q(D_Q)$ assure a *deterministic* accuracy bound η:

(a) for *each* approximate answer $s \in Q(D_Q)$, there exists an exact answer $t \in Q(D)$ that is η-close to s, *i.e.,* s is within distance η of t; and

(b) for *each* exact answer $t \in Q(D)$, there exists an approximate answer $s \in Q(D_Q)$ that is η-close to t.

That is, $Q(D_Q)$ includes only "relevant" answers, and "covers" all exact answers. It finds sensible answers in users' interest, and suffices for exploratory queries, *e.g.,* real-time problem diagnosis on logs [2].

The objective is ambitious. As observed in [9], approximate query answering is challenging. Previous approaches often adopt an *one-size-fit-all* synopsis D_c and computes $Q(D_c)$ for all queries Q posed on D. The approaches "substantially limit the types of queries they can execute" [2], and often focus on aggregate queries (max, min, avg, sum, count). Moreover, they make various assumptions on future queries, *i.e.,* workloads, query predicates or QCSs, *i.e.,* "the frequency of columns used for grouping and filtering does not change over time". Worse still, they provide either no accuracy guarantee at all, or probabilistic error rates for aggregate queries only. Such error rates do not tell us how "good" each approximate answer is.

Nonetheless, data-driven approximation is feasible under access schema.

Example 2: Continuing with Example 1, consider query Q_2 to *find me hotels that cost at most \$95 per night and are in a city where one of my friends lives*:

```
select h.address, h.price
from   poi as h, friend as f, person as p
where  f.pid = $p_0$ and f.fid = p.pid and p.city = h.city
       and h.type = "hotel" and h.price $\leq$ 95
```

We can compute $Q_2(D_0)$ in a big dataset D_0 of trillions of tuples given a small α, *e.g.,* 10^{-4}, *i.e.,* when our available resources can afford to access at most $10^{-4} * |D_0|$ tuples. This is doable by using an access schema $\mathcal{A}_0$, which includes φ_1 and φ_2 of Example 1 and in addition, the following extended access constraints:

- ψ_1: $\mathsf{poi}(\{\mathsf{type}, \mathsf{city}\} \to \{\mathsf{price}, \mathsf{address}\}, 1, (e_p^1, e_a^1))$,

  ...
- ψ_m: $\mathsf{poi}(\{\mathsf{type}, \mathsf{city}\} \to \{\mathsf{price}, \mathsf{address}\}, 2^m, (e_p^m, e_a^m))$, where $m = \lceil \log_2 M \rceil$.

Here M is the maximum number of distinct poi tuples in D_0 grouped by (type, city). We build an index for each ψ_i in $\mathcal{A}_0$ such that for $i \in [1, m]$, given any (type, city)-value (c_t, c_c), we can retrieve a set T of at most 2^i (price, address) values from D_0 by using the index for ψ_i; moreover, for each poi tuple (c'_a, c_t, c_c, c'_p) in D_0, there exists $(c_p, c_a) \in T$ such that the (price, address)-value (c'_p, c'_a) differs from (c_p, c_a) by distance at most (e_p^i, e_a^i). That is, T represents (price, address) values that correspond to (c_t, c_c) with at most 2^i tuples, subject to distances (e_p^i, e_a^i). Intuitively, the indices give a hierarchical representation of relation poi with different resolutions $i \in [1, m]$. The higher the resolution i is, the smaller the distance (e_p^i, e_a^i) is, and the more accurate the index for ψ_i represents D_0.

Assume $\alpha|D_0| > 10000$, as in Facebook dataset D_0. Then under $\mathcal{A}_0$, we can find hotels by accessing at most $\alpha|D_0|$ tuples as follows: (a) fetch a set T_1 of fid's with p_0 by accessing at most 5000 friend tuples using φ_1; (b) for each fid in T_1, fetch 1 associated city with φ_2, yielding a set T_2 of at most 5000 city values; (c) for each city c in T_2, fetch at most 2^{k_α} (price, address) pairs corresponding to ("hotel", c) by using ψ_{k_α}, where $k_\alpha = \lfloor \log_2(\alpha|D_0| - 10000) \rfloor$; and (d) return a set S of those (price, address) values with price at most $(95 + e_p^{k_\alpha})$, as approximate answers to Q_2 in D_0. The process accesses at most $5000 + 5000 + 2^{k_\alpha} \leq \alpha|D_0|$ tuples in total.

The set S of answers is accurate: (1) for each hotel $h_0(c_p, c_a)$ in the exact answers $Q(D_0)$, there exists (c'_p, c'_a) in S that are within $e_p^{k_\alpha}$ and $e_a^{k_\alpha}$ of c_p and c_a, respectively; and (2) for each hotel $h'(c'_p, c'_a)$ in S, its price c'_p exceeds 95 by at most $e_p^{k_\alpha}$, *e.g.,* $e_p^{k_\alpha} = 4$ and $c'_p = 99$, and c'_a is the address of hotel h'. Moreover, the larger α is, the smaller $e_p^{k_\alpha}$ and $e_a^{k_\alpha}$ are, and the more accurate S is. $\square$

It has been shown [4] that for any dataset D, there exists such an access schema $\mathcal{A}$ such that D conforms to $\mathcal{A}$, and for any resource ratio $\alpha \in (0, 1]$ and SQL queries Q over D, aggregate or not, there exists a dataset $D_Q \subseteq D$ identified by reasoning about $\mathcal{A}$ and a deterministic accuracy bound η such that $|D_Q| \leq \alpha|D|$ and $\mathsf{accuracy}(Q(D_Q), Q, D) \geq \eta$. Moreover, the larger α is, the higher η is.

As opposed to previous approximate query answering approaches, the data-driven approximation scheme is able to answer SQL queries Q that are (1) *unpredictable*, *i.e.,* without assuming any prior knowledge about Q, (2) and *generic*, *aggregate or not*, with (3) *deterministic* accuracy η in terms of both relevance and coverage.

We find that the approximation scheme computes approximate answers to SQL queries, aggregate or not, with accuracy $\eta \geq 0.82$ for SQL queries, even when α is as small as 5.5×10^{-4} [4]. That is, it reduces D of PB size to D_Q of 550GB.

4 A Resource Bounded Query Evaluation Framework

We are now ready to present BEAS (Boundedly EvAluable Sql), a resource-bounded framework for querying big relations. For a big dataset D in an application, BEAS takes a *resource ratio* $\alpha \in (0, 1]$ as a parameter, and discovers an access schema $\mathcal{A}$. Given an SQL query Q posed on D, BEAS works as follows:

(1) it checks whether Q is boundedly evaluable under $\mathcal{A}$, *i.e.,* exact answers $Q(D)$ can be computed by accessing $D_Q \subseteq D$ such that $|D_Q|$ is independent of $|D|$;
(2) if so, it computes $Q(D)$ by accessing a bounded fraction D_Q of D;
(3) otherwise, BEAS identifies D_Q with $|D_Q| \leq \alpha|D|$, and computes $Q(D_Q)$ with a deterministic accuracy bound η, based on data-driven approximation.

That is, under the resource constraint α, BEAS computes exact answers $Q(D)$ when possible, and approximate answers $Q(D_Q)$ otherwise with accuracy η.

As opposed to conventional DBMS, BEAS is unique in its ability to (1) comply with *resource ratio* α, *i.e.,* it can scale with arbitrarily large datasets D by adjusting α based on available resources, (2) decide whether Q is *boundedly evaluable*, (3) answer *unpredictable* and *generic* SQL queries, *aggregate or not*, with *deterministic* accuracy η, and (4) be plugged into commercial DBMS and provide the DBMS with an immediate capacity to query big relations under constrained resources.

In light of these, BEAS is promising for providing small companies with the capability of big data analytics and hence, to benefit from big data services. It can also help big companies such as Huawei to reduce the cost and improve efficiency [22]. Moreover, parallel processing is not a silver bullet for big data analytics. Indeed, one might expect a parallel algorithm to have the *parallel scalability*, *i.e.,* the algorithm would run faster given more processors. However, few algorithms in the literature have this performance guarantee. Worse yet, some computation problems are not parallel scalable, *i.e.,* there exist no algorithms for them such that their running time can be substantially reduced by adding processors, no matter how many processors are used [13, 23]. For such computation problems, bounded evaluation and data-driven approximation offer a feasible solution.

The idea of resource-bounded query answering is not limited to relations. It has been shown that bounded evaluation improves the performance of graph pattern matching via subgraph isomorphism, an intractable problem [20] that is widely used in social media marketing [15, 16] and knowledge base expansion [17], by 4 orders of magnitude on average [6]. For personalized social search via subgraph isomorphism, data-driven approximation retains 100% accuracy (*i.e.,* $\eta = 1$) when α is as small as $1.5 * 10^{-6}$ [14], *i.e.,* when processing graphs G of 1PB, they access only 15GB of data, *i.e.,* reducing G from PB to GB while retaining high accuracy!

Acknowledgements Fan is supported in part by ERC 652976, 973 Program 2014CB340302, NSFC 61421003, EPSRC EP/M025268/1, Beijing Advanced Innovation Center for Big Data and Brain Computing, and Shenzhen Peacock Program 1105100030834361.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S., Stoica, I.: BlinkDB: Queries with bounded errors and bounded response times on very large data. In: EuroSys (2013)
3. Cao, Y., Fan, W.: An effective syntax for bounded relational queries. In: SIGMOD (2016)
4. Cao, Y., Fan, W.: Data driven approximation with bounded resources. PVLDB **10**(9), 973–984 (2017)
5. Cao, Y., Fan, W., Geerts, F., Lu, P.: Bounded query rewriting using views. In: PODS (2016)
6. Cao, Y., Fan, W., Huang, R.: Making pattern queries bounded in big graphs. In: ICDE (2015)
7. Cao, Y., Fan, W., Wang, Y., Yuan, T., Li, Y., Chen, L.Y.: BEAS: bounded evaluation of SQL queries. In: SIGMOD, pp. 1667–1670 (2017)
8. Cao, Y., Fan, W., Wo, T., Yu, W.: Bounded conjunctive queries. PVLDB (2014)
9. Chaudhuri, S., Ding, B., Kandula, S.: Approximate query processing: No silver bullet. In: SIGMOD, pp. 511–519 (2017)
10. Facebook: Introducing Graph Search. *https://en-gb.facebook.com/about/graphsearch* (2013)
11. Fan, W., Geerts, F., Cao, Y., Deng, T.: Querying big data by accessing small data. In: PODS (2015)
12. Fan, W., Geerts, F., Libkin, L.: On scale independence for querying big data. In: PODS (2014)
13. Fan, W., Wang, X., Wu, Y.: Distributed graph simulation: Impossibility and possibility. PVLDB **7**(12) (2014)
14. Fan, W., Wang, X., Wu, Y.: Querying big graphs within bounded resources. In: SIGMOD (2014)
15. Fan, W., Wang, X., Wu, Y., Xu, J.: Association rules with graph patterns. PVLDB **8**(12), 1502–1513 (2015)
16. Fan, W., Wu, Y., Xu, J.: Adding counting quantifiers to graph patterns. In: SIGMOD, pp. 1215–1230 (2016)
17. Fan, W., Wu, Y., Xu, J.: Functional dependencies for graphs. In: SIGMOD (2016)
18. Grujic, I., Bogdanovic-Dinic, S., Stoimenov, L.: Collecting and analyzing data from e-government Facebook pages. In: ICT Innovations (2014)
19. ITPro Newsletter: Samsung reveals 'largest ever' SSD at 15TB. *http://www.itpro.co.uk/ssds/26175/samsung-reveals-largest-ever-ssd-at-15tb* (2016)

20. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
21. Stack Overflow: SQL: Inner joining two massive tables. *http://stackoverflow.com/questions/1750001/sql-inner-joining-two-massive-tables*
22. University of Edinburgh: Huawei deal to advance expertise in data science. *http://www.ed.ac.uk/news/2017/huawei-deal-to-advance-expertise-in-data-science* (2017)
23. Xie, C., Chen, R., Guan, H., Zang, B., Chen, H.: SYNC or ASYNC: time to fuse for distributed graph-parallel computation. In: PPoPP, pp. 194–204 (2015)