EmptyHeaded: A Relational Engine for Graph Processing

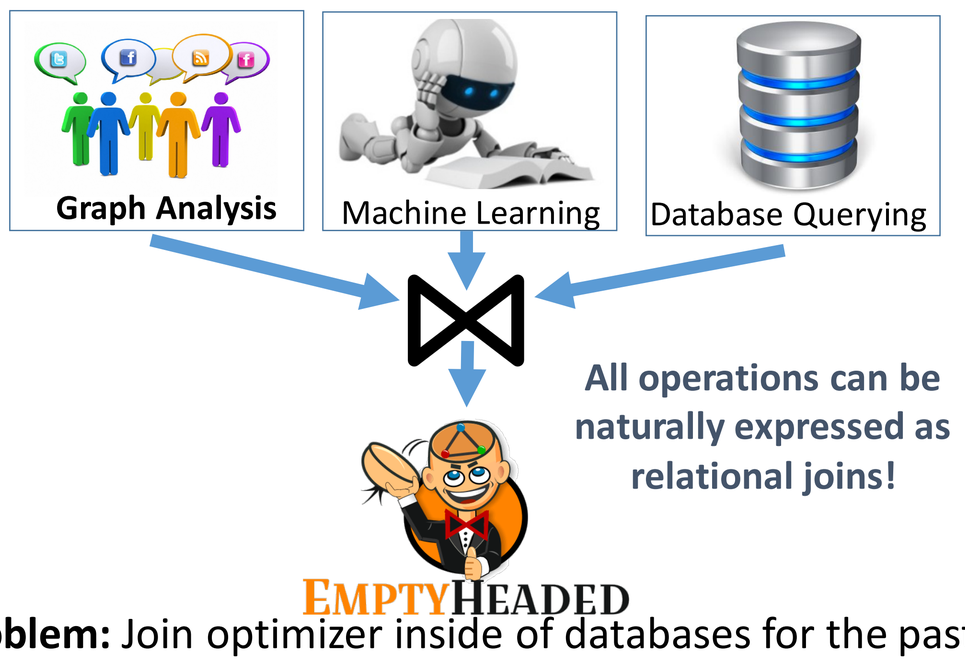
Stanford University

Christopher Aberger, Susan Tu, Kunle Olukotun, and Christopher Ré

caberger@stanford.edu

Overview

- **Goal:** Can we express **graph analysis**, linear algebra, and classic querying in a **single RDBMS engine**?



- **Problem:** Join optimizer inside of databases for the past 40 years can be asymptotically suboptimal!
- **Solution:** Worst-case optimal joins [Ngo et al. 2012] !

Joins Algorithms

Traditional Databases: Compute joins in a **pairwise** fashion over **relations**.

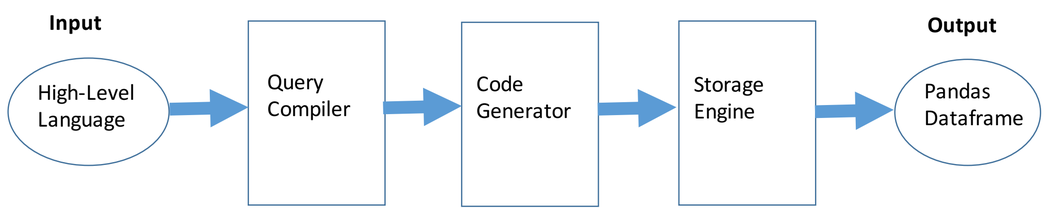
Worst-Case Optimal Joins: Compute joins in a **multiway** fashion over **attributes**.

Example: Find all 3-cliques in a telephone network.

Traditional Database: $O(N^2)$

Worst-Case Optimal Join: $O(N^{3/2})$

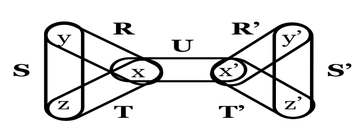
System Overview



Query Compiler

- Replace relational algebra with *generalized hypertree decompositions* (GHDs) to always guarantee **tight** theoretical running times!

```
select cluster
from social_network
where ...
```



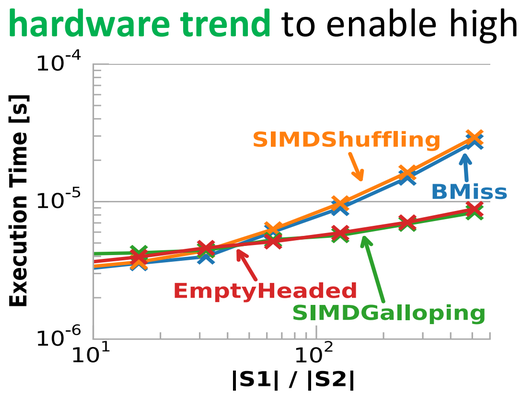
Aggregations

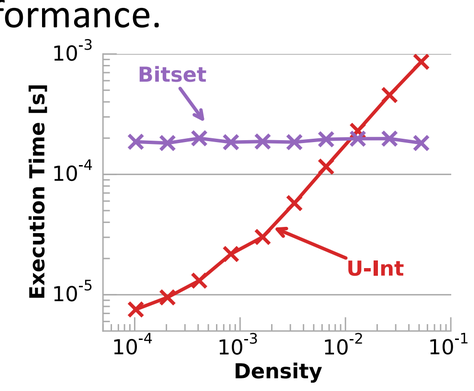
- Computing aggregations via **annotations** over **semi-rings** enables us to express linear algebra applications with tight theoretical bounds as well! [Puttagunta PODS '16]

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ c_0 \\ a_1 \\ c_1 \\ a_2 \\ c_2 \\ a_3 \\ c_3 \end{bmatrix} \Leftarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Storage Engine

- **Automatically** decide **algorithms** and **layouts** designed for **SIMD hardware trend** to enable high performance.





Results

Triangle Query

Dataset	EmptyHeaded	PowerGraph	Snap	LogicBlox
Google+	0.31s	8.40x	4.18x	83.74x
LiveJournal	0.48s	5.17x	10.72x	23.53x
Orkut	2.36 s	2.94x	4.09x	19.24 x
Twitter	56.81s	4.40x	2.22x	30.60x

We benchmarked standard relational databases such as PostGreSQL and MonetDB (and even SparkSQL) and they were all at least 2 orders of magnitude slower than EmptyHeaded!

Barbell Query

Dataset	EmptyHeaded	-GHD	-Layout	LogicBlox
Google+	3.17s	t/o	1.14x	t/o
LiveJournal	1.67s	t/o	344.90x	t/o
Orkut	8.87s	t/o	47.81x	t/o

t/o indicates that the query ran for over 30 minutes.

Beyond Classic Join Queries

With a couple straightforward additions to our join optimizer (recursion and aggregations) our approach can be competitive on queries outside of the scope of classical database queries.

PageRank Query (5 iterations)

Dataset	EmptyHeaded	Galois	PowerGraph	LogicBlox
Google+	**0.10s**	0.021s	0.24s	7.03s
LiveJournal	**0.58s**	0.51s	4.32s	25.03s
Orkut	**0.65s**	0.59s	4.48s	75.11s
Twitter	**15.41s**	17.98s	57.00	442.85s

SSSP

Dataset	EmptyHeaded	Galois	PowerGraph	LogicBlox
Google+	0.024s	**0.008s**	0.22s	41.81s
LiveJournal	0.19s	**0.062s**	1.80s	102.83s
Orkut	0.24s	**0.079s**	2.30s	215.25s
Twitter	7.87s	**2.52s**	36.90s	379.16s

Future Work

(1) Classic Query Workloads (TPC-H)

(2) Machine Learning (matrix multiply)

Make linear algebra + querying a reality!

More Information

Open Source Repository

`https://github.com/HazyResearch/EmptyHeaded`

Includes iPython Notebook tutorials!

Contact Information (Student)

`www.stanford.edu/~caberger`