Correlated Sample Synopsis On Big Data

David Wilson

Youngstown State University, MCIS
Dr. Feng Yu

Introduction

1 Topic

Correlated Sample Synopses (CS2)

- Statistical Summary for query optimization in databases
- Through methods, including SRSWOR, provides unbiased, fast size estimations for all queries with joins and arbitrary selections, with high precision.

Introduction

2 Motivation

CS2 has not been tested on big data

- Will CS2 maintain its speed increase when performing search query estimations on big datasets?
- Will CS2 maintain accuracy on big datasets when performing search query estimations?

Introduction

4 Objectives

- Survey research on topics
  - Big Data
  - Apache Hadoop
  - Apache Hive
  - Join Synopses

Introduction

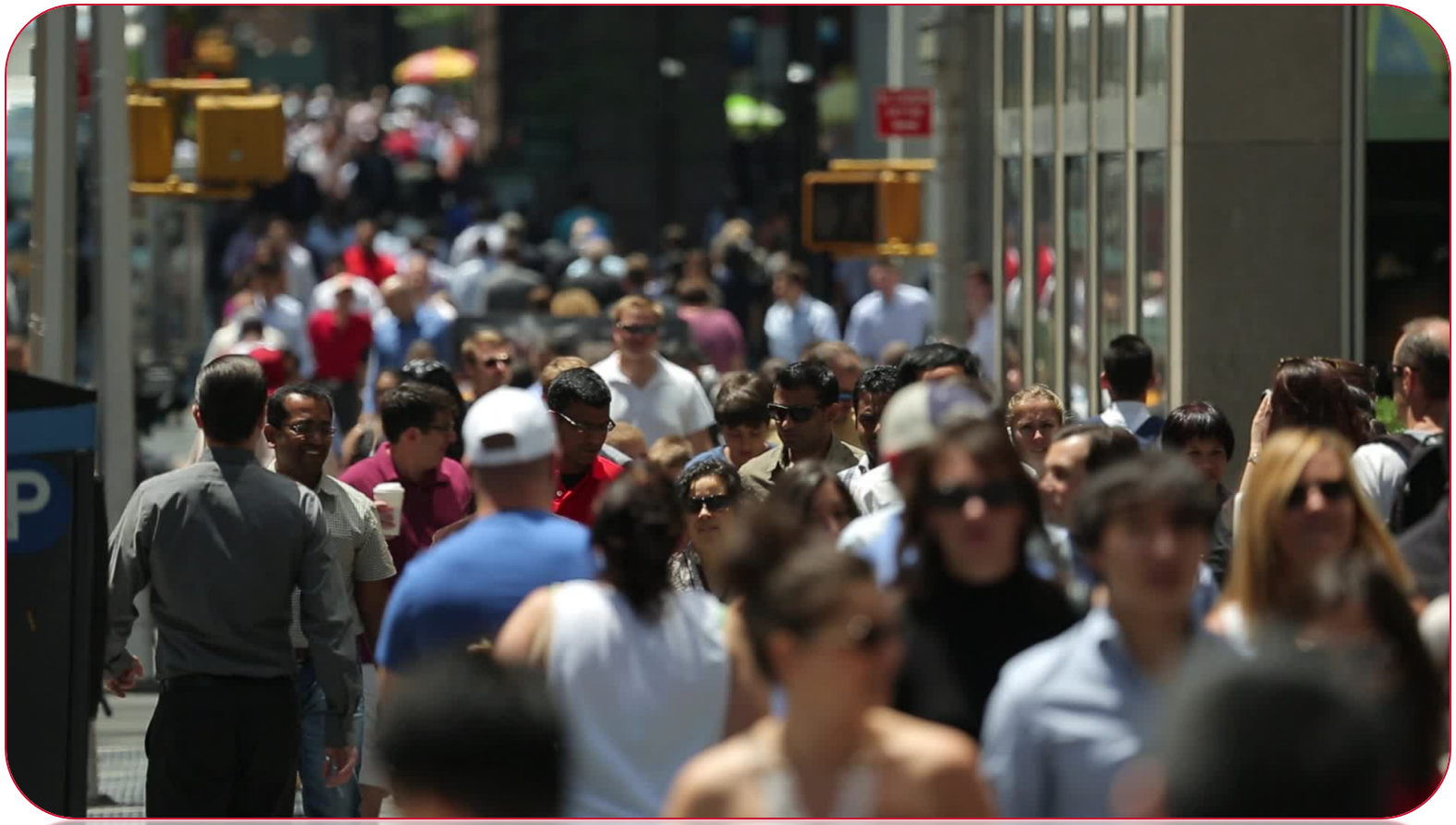
4 Objectives

- Extend correlated sampling and query estimation to big data including
  - Join Graphs
  - SRSWOR
  - CS2 Construction

Background: Big Data

Data Creation

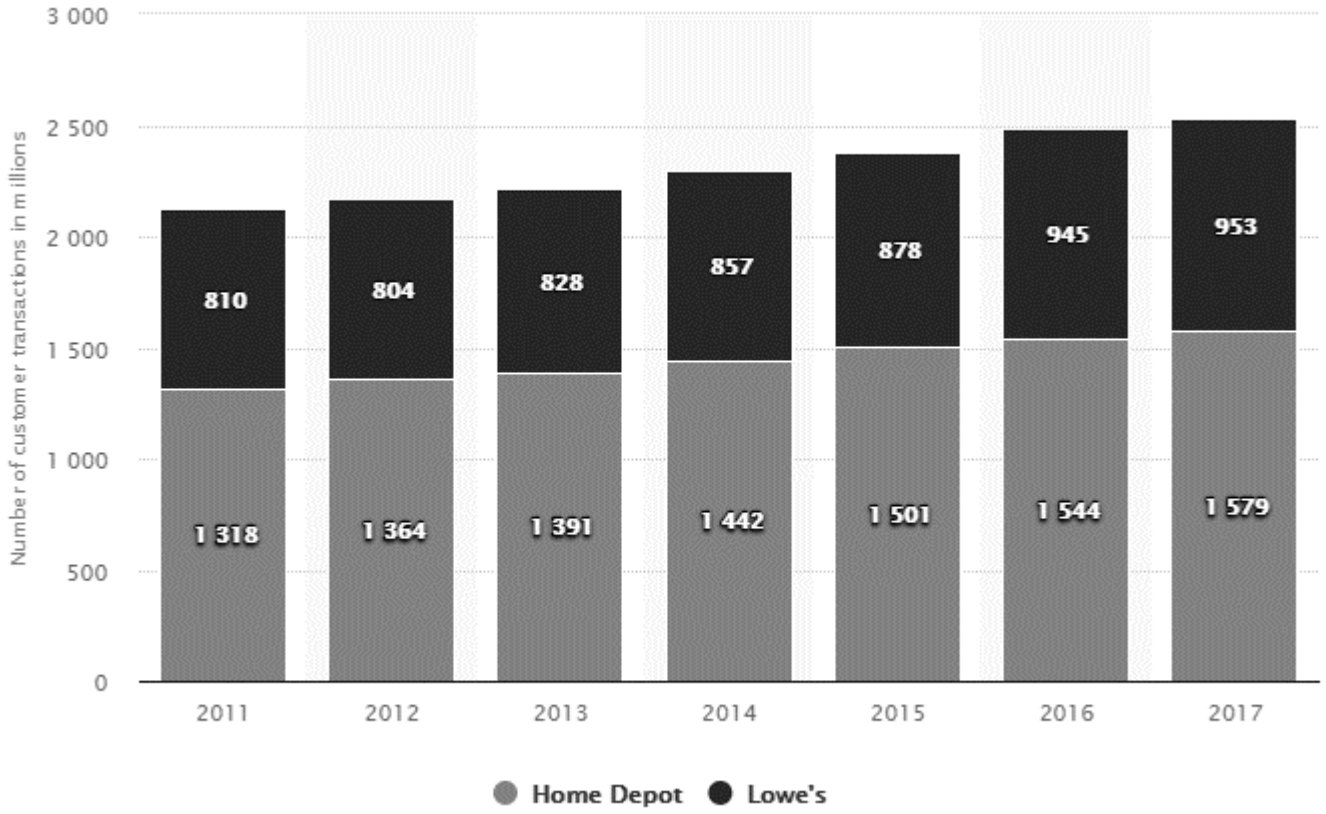
- 2.5 Quintillion Bytes (2.5 Billion Gigabytes) Daily[1]
- 90% of all data created past two years



Background: Big Data

- Mistaken as only large amounts of data
- First coined by John Mashey of Silicon Graphics in 1998 (Debated)
- Not until 2001 that Big Data took the form of what it is considered today, when Doug Laney created the Three V's[2]
  - Volume
  - Variety
  - Velocity

Background: Big Data



Lowes [5]

- 953 Million Transactions
  - 1813 Transactions/Min

Home Depot

- 1.579 Billion Transactions
  - 3004 Transactions/Min

Background: Apache Hadoop

Hadoop

- Created in 2005 by Doug Cutting and Mike Cefarella [6]
- Open Source distributed processing framework that manages data processing and provides storage for big data applications running in clustered systems
- Framework consists of multiple modules
  - Hadoop Common
  - Hadoop Distributed File System
  - Hadoop Yarn
  - Hadoop MapReduce

Background: Apache Hadoop

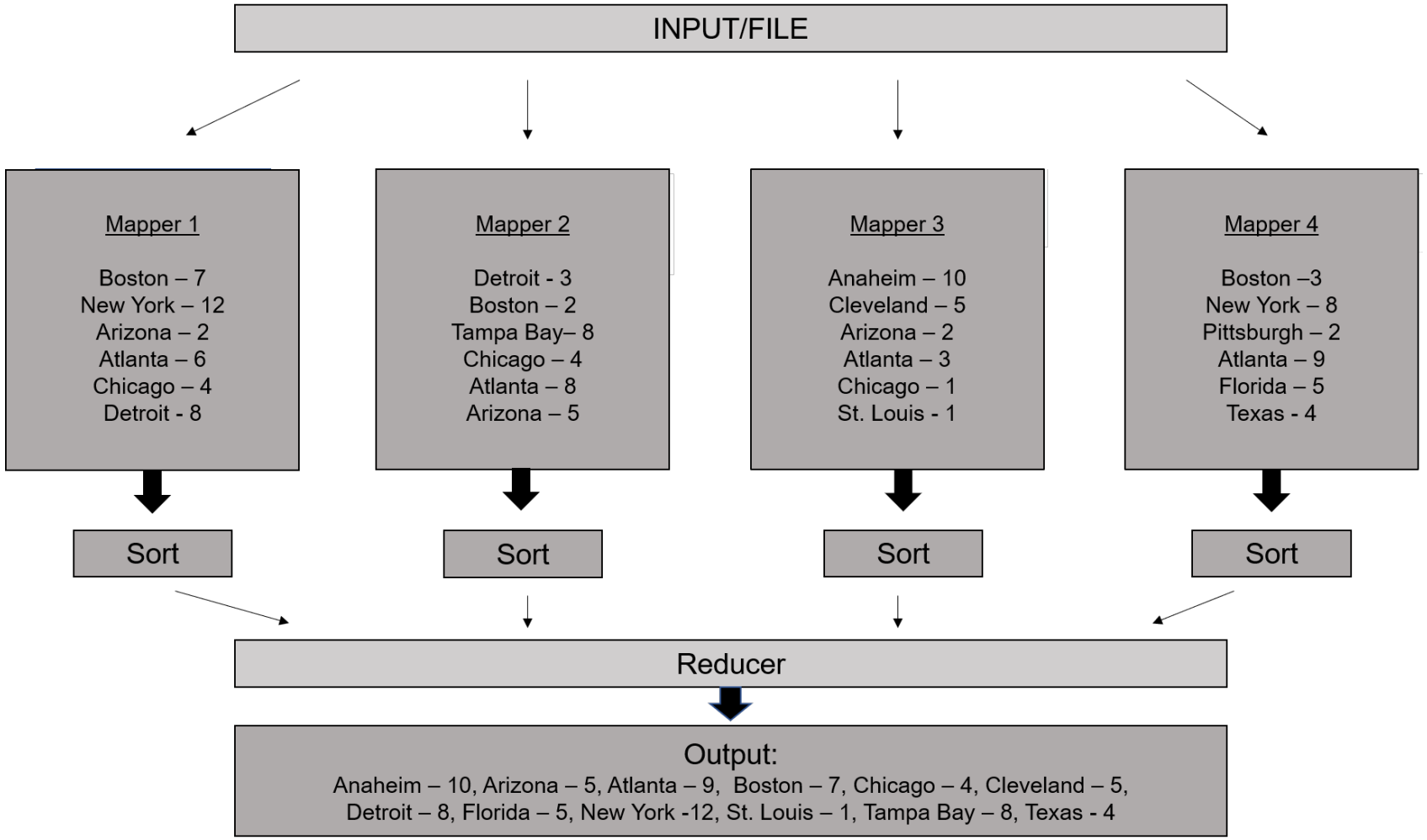
Hadoop Common[7]

- Storehouse for other Hadoop modules
  - Holds all files that other Hadoop Modules need to run

Hadoop Distributed File System (HDFS)

- Deals with storage of data
- Built to combat hardware failure by using a process called replication
  - Name Node - holds all meta data
  - Data Node – holds all actual data
    - Consists of a multitude of blocks
    - Each block stored in three different data node locations in cluster
    - Node/Machine/Cluster failure block copy is made on another Node/Machine/Cluster

Background: Apache Hadoop



Background: Apache Hive

Hadoop Hive [8]

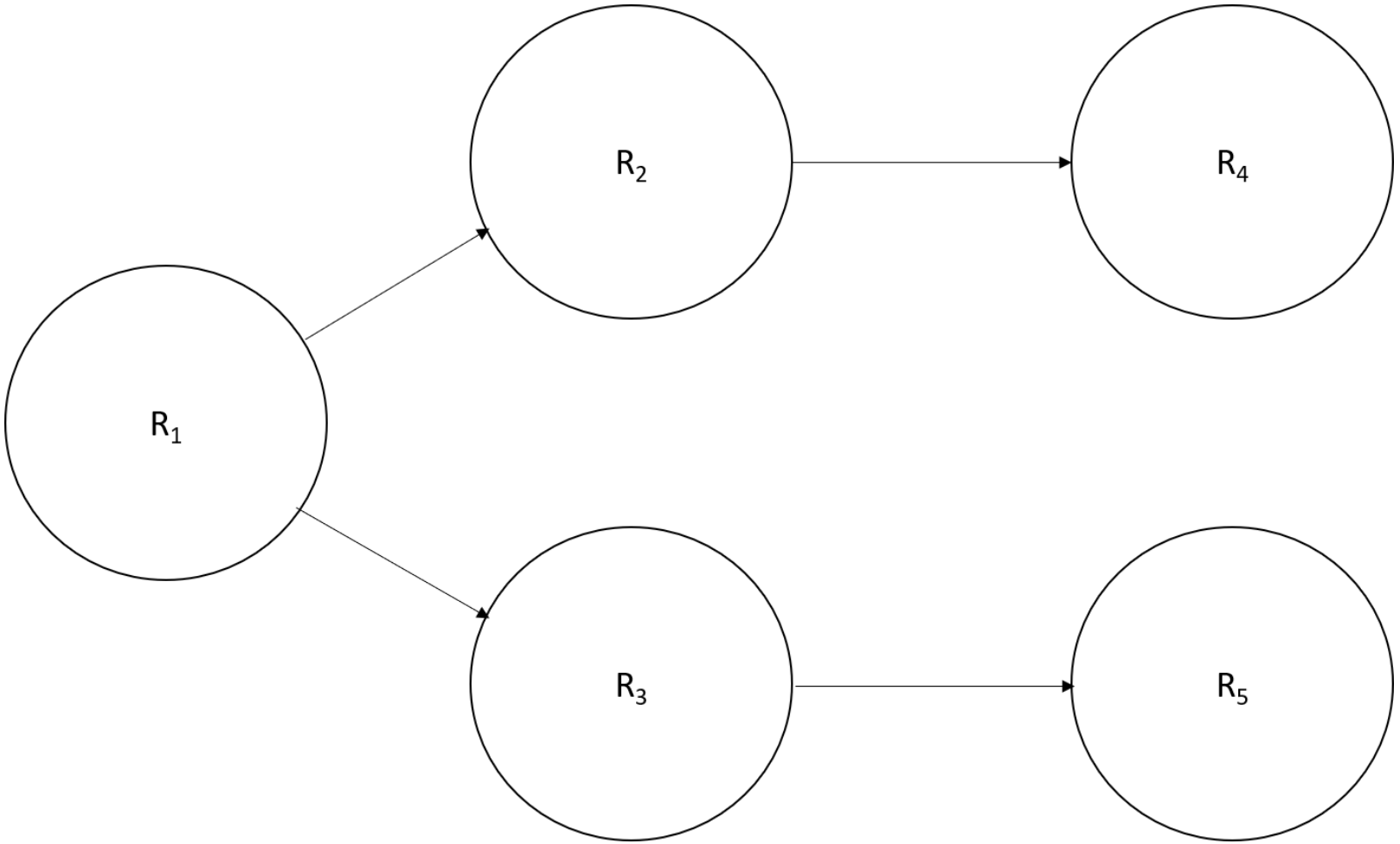
- Open Source tool
- Built by developers at Facebook in 2007
- Runs on Apache Hadoop framework
  - Allows developers to use HQL (Hive Query Language) on HDFS and MapReduce
    - HQL similar to SQL
    - Allows users to summarize and analyze data using traditional command line or GUI HWI(Hive Web Interface)

Background: Join Graphs

Join Graph[9]

- A visual representation of a database in which the flow of joins is explained
  - Can be created to take into consideration the relational join types (many-to-one, one-to-many, many-to-many)
- A relation is considered joinable between R_i and R_k, $i \neq k$ when there is a path ≥ 1 between the relations R_i and R_k

Background: Join Graphs



A Basic Join Graph

Background: SRSWOR

SRSWOR (Simple Random Sample Without Replacement)

- Tested as a sample synopsis
  - Is unbiased due to its nature
  - An estimation of a single relation can be made with SRSWOR
  - Joinable Query Estimations cannot be made with SRSWOR by itself
    - An individual SRSWOR on one relation does not correlate with a SRSWOR on another relation

Background: SRSWOR

Customer

CustID	Fname	Lname	Age
1	Steve	Jones	30
2	Rob	Mccarthy	22
3	Lisa	Roben	36
4	Calvin	Obernick	48
5	Shawn	Tucsan	19
6	Sherry	Dobbs	25
7	Kelly	Harper	29
8	John	Robeck	54
9	Allison	Calvon	26
10	Jenna	Sharon	28

Orders

OrderID	CustID	City	State
1	4	Oakland	CA
2	7	Chicago	IL
3	2	Pittsburgh	PA
4	8	Tampa	FL
5	1	Atlanta	GA
6	7	Chicago	IL
7	10	Seattle	WA
8	3	Columbus	OH
9	5	Houston	TX
10	9	Buffalo	NY

SRSWOR at 20% of Relation Size

9	**Allison**	**Calvon**	**26**
3	**Lisa**	**Roben**	**36**

Join

SRSWOR at 20% of Relation Size

1	**4**	**Oakland**	**CA**
8	**3**	**Columbus**	**OH**

JOIN RESULT

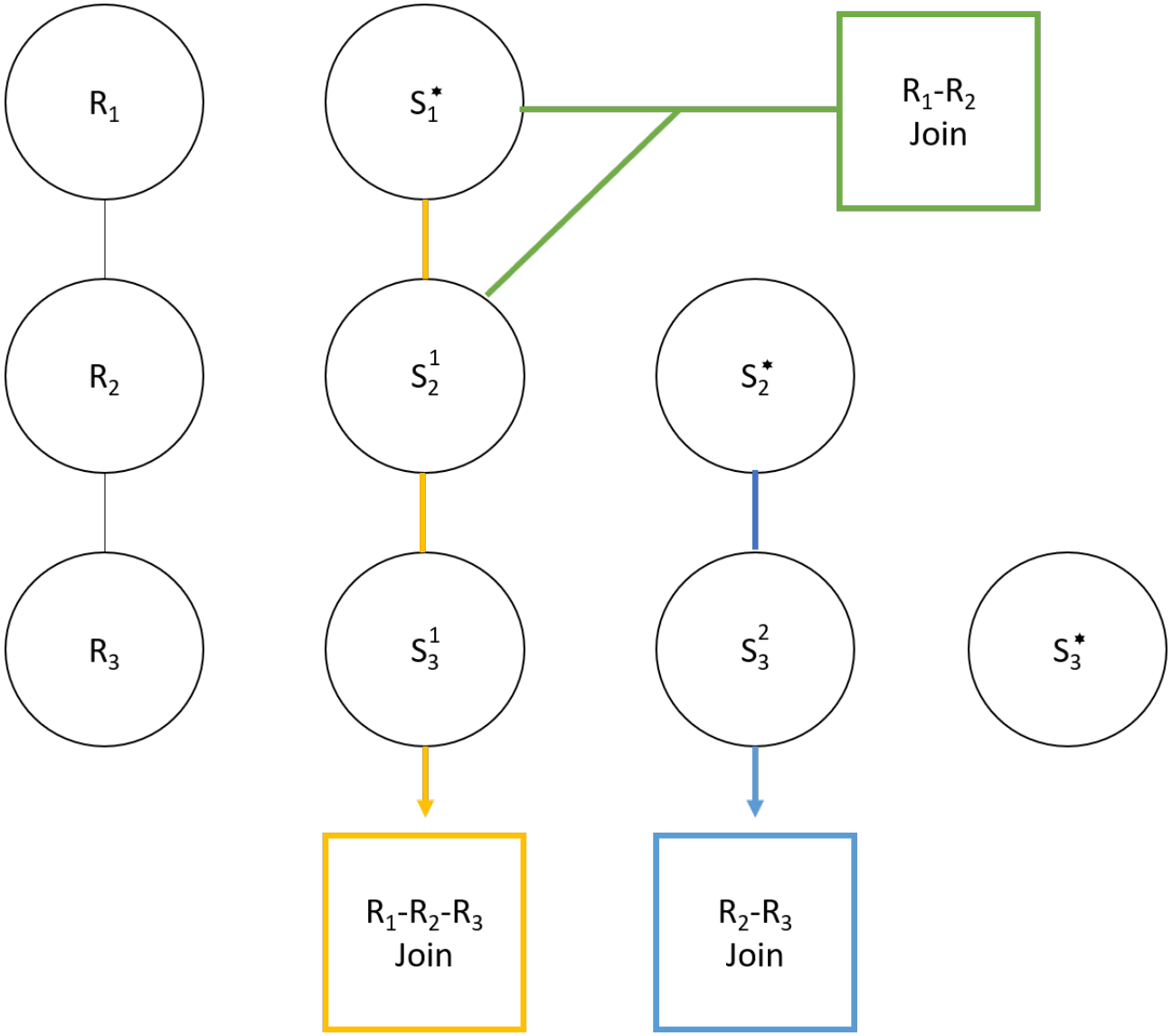
3	**Lisa**	**Roben**	**36**	**8**	**3**	**Columbus**	**OH**

Background: Join Synopses

Join Synopsis [10]

- Uses SRSWOR
  - Process it uses adds correlation between individual relations
    - Uses foreign key joins
    - Computes small samples of a small set of joins
    - Samples stored
    - Samples joined with individual SRSWOR to form a unbiased finalized set of correlated tuples that can be used for query estimation
- Due to foreign key joins can only do many-to-one, and one-to-many relations (cannot do many-to-many)
- Query Join Size estimation works well, BUT, storage grows exponentially as size of database gets larger

Background: Join Synopses

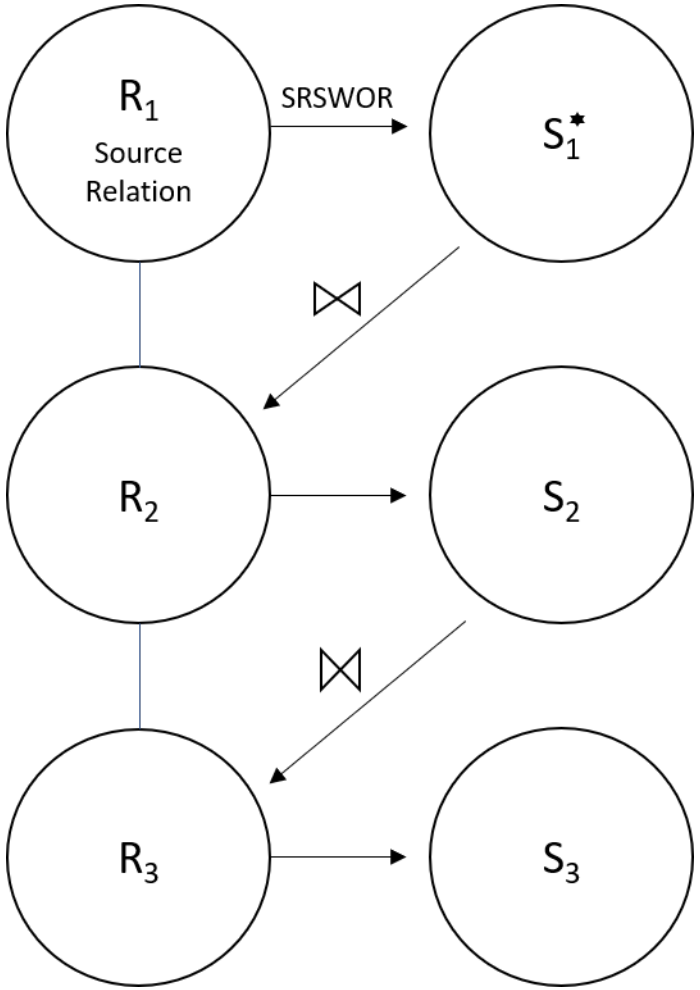


CS2 - Construction

CS2 Construction[9]

- A join graph of the database must be created and a source relation must be chosen
  - The source relation is preferred to be a relation that reaches the most other relations
  - Suggested to use a many-to-one relationship, as one-to-many and many-to-many can cause the synopsis to grow considerably, subtracting from the overall number of sample tuples that can be taken from the source relation

CS2 - Construction



CS2 - Construction

Algorithm 1 Correlated Sampling

Require: G — Join Graph of the Database; n_1 — Sample Size

1: R_a=Source_Selection(G) //source relation selection
2: $S_a^* = \emptyset$; $S_i = \emptyset$ $(\forall\ i \neq a)$
3: $S_a^* = \text{SRSWOR}(R_a, n_1)$ //collect simple random sample
4: $W = \{R_a\}$ //mark R_a as visited
5: **while** $\exists$ unvisited edge $\langle R_i, R_j \rangle$ with $R_i \in W$ **do**
6: $S_j = S_j \cup \{t | t \in R_j, t$ is directly joinable with a tuple in $S_i\}$ //collect correlated sample tuples
7: $W = W \cup \{R_j\}$ //mark R_j as visited
8: **end while**
9: $\mathbb{S} = \{S_a^*\} \cup (\cup_{j \neq a}\{S_j\})$
10: **return** $\mathbb{S}$ — Correlated Sample Synopsis

CS2 – Query Estimation

Query Estimation

- Taking results from sample query, to be used to estimate source query result sizes
  - Source Query (Used in thesis)
    - Includes the source relation in the join search query estimation
    - No additional steps must be taken to perform a join query estimation
    - Formula for Query Estimation
      Sample tuple size / Sample Ratio
    - Formula for Relative Error
      *ABS(Estimation -Source Count)/Source Count * 100%*

CS2 – Query Estimation

No-Source Query (beyond scope of thesis)

- Does not include a source relation in the join search query
- Without source relation, the relation with the least index, based off sampling order, would be considered the highest relation
- Without SRSWOR additional steps are needed
  - Joinable Tuple Sampled Ratio, or JR, is a procedure of backtracking to the source relation in a no source query (reverse sampling), and supplying it with the ability to estimate the join query size. The process is called reverse estimation.

Experiment – Setup

Setup

- Cluster of five nodes on a remote server (Sarah Cloud in Data Lab) are created
  - Master Node 1
    - 4 Intel Xeon CPU's (E5-2630 v4 @ 2.20 GHz)
    - 16 GB RAM
  - Master Node 2
    - 2 Intel Xeon CPU's (E5-2630 v4 @ 2.20 GHz)
    - 10 GB RAM
  - Three Worker Nodes
    - Intel Xeon CPU's (E5-2630 v4 @ 2.20 GHz)
    - 8 GB RAM

Experiment – Setup

```
hive> show databases;
OK
default
s_tpch10g
s_tpch1g
tpch10g
tpch1g
Time taken: 0.018 seconds, Fetched: 5 row(s)
hive> use s_tpch10g;
OK
Time taken: 0.243 seconds
hive>  select count (*) from lineitem,orders,customer where l_orderkey = o_orderkey and o_custkey = c_custkey and c_acctbal > 500;
Query ID = dswilson_20181203141757_fe14c352-1057-4723-9647-d69bb0a73524
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1537548368557_0037)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      8          8        0        0       0       0
Map 3 ..........   SUCCEEDED      6          6        0        0       0       0
Map 4 ..........   SUCCEEDED      6          6        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 12.21 s
--------------------------------------------------------------------------------
OK
516682
Time taken: 13.545 seconds, Fetched: 1 row(s)
hive>
```

Experiment – Setup

- Server
  - Running Apache Hive with Apache Hive setup
  - Connected through PUTTY SSH using Cisco AnyConnect VPN
  - Unit used for connection
    - Intel Core i7 – 5500U CPU @ 2.40GHz
    - 12 GB RAM

Experiment – Datasets

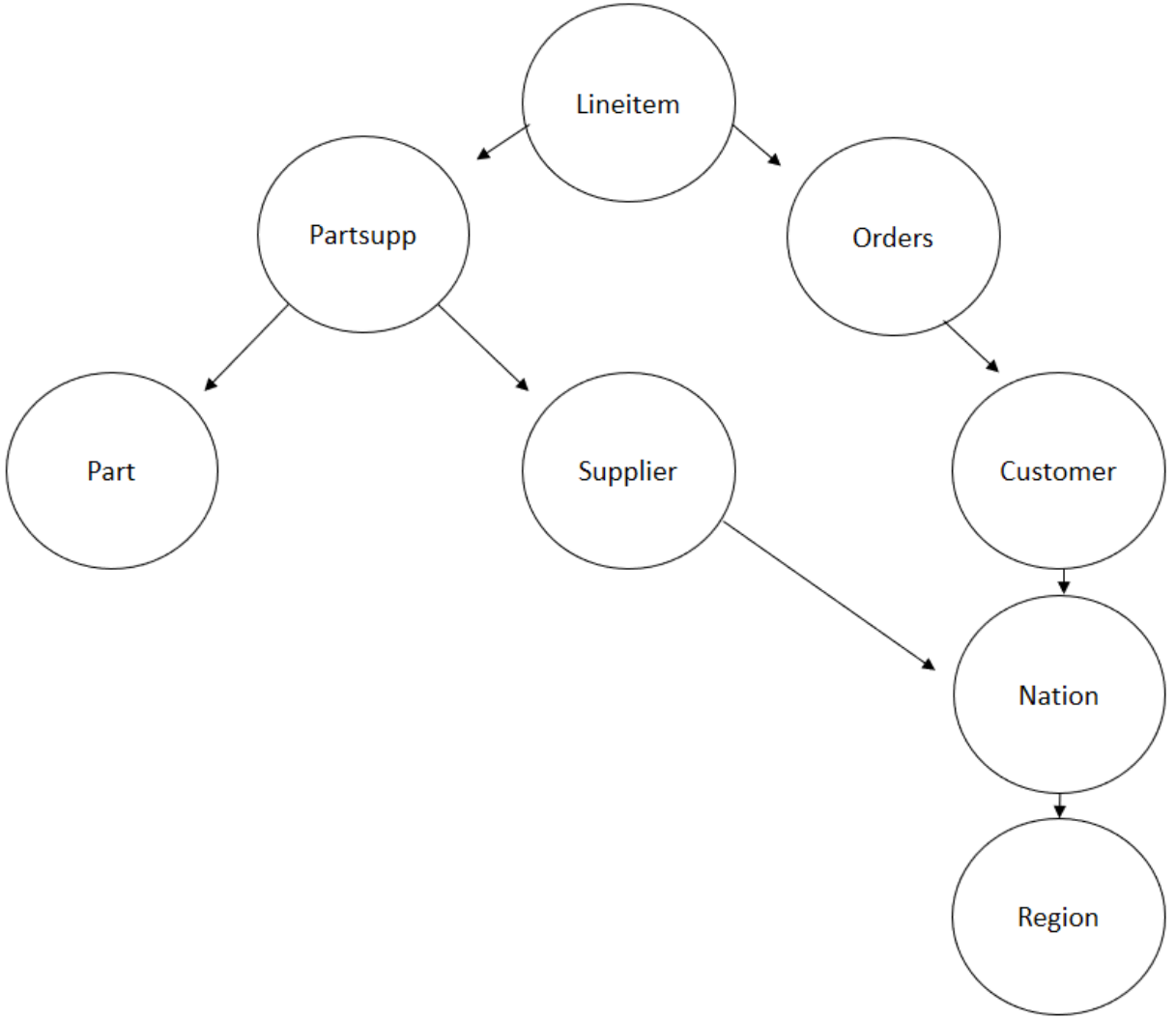
- Two datasets used
  - Both created using TPC-H benchmark
  - First dataset is 1GB
  - Second dataset is 10GB
  - Each dataset holds eight relations
    - Lineitem, Customer, Orders, Partsupp, Part, Supplier, Nation, and Region

Experiment

Step 1 – Using source dataset, source relation as well as join graph path is chosen.

- Lineitem is the source relation. It holds the most many-to-one relationships and connects with the most other relations
- Join Graph Path is as follows
  - Lineitem -> Orders, Lineitem-> Partsupp, Orders -> Customer, Partsupp -> Part, Partsupp -> Supplier, Customer -> Nation, Nation -> Region

Experiment



TPC-H Join Graph

Experiment

Step 2 – a empty set must be created to store samples of source dataset. The datasets are denoted as

- Source – tpch1g and tpch10g
- Sample –s_tpch1g and s_tpch10g

Step 3 – before creating the SRSWOR a sample dataset size must be selected. The decision is decided that it will be one percent of the source dataset

- HQL Code used to create SRSWOR
  - create table s_tpch10g.lineitem as select * from tpch10g.Lineitem
    where rand () <= 0.01
    Distribute by rand ()
    Sort by rand ();

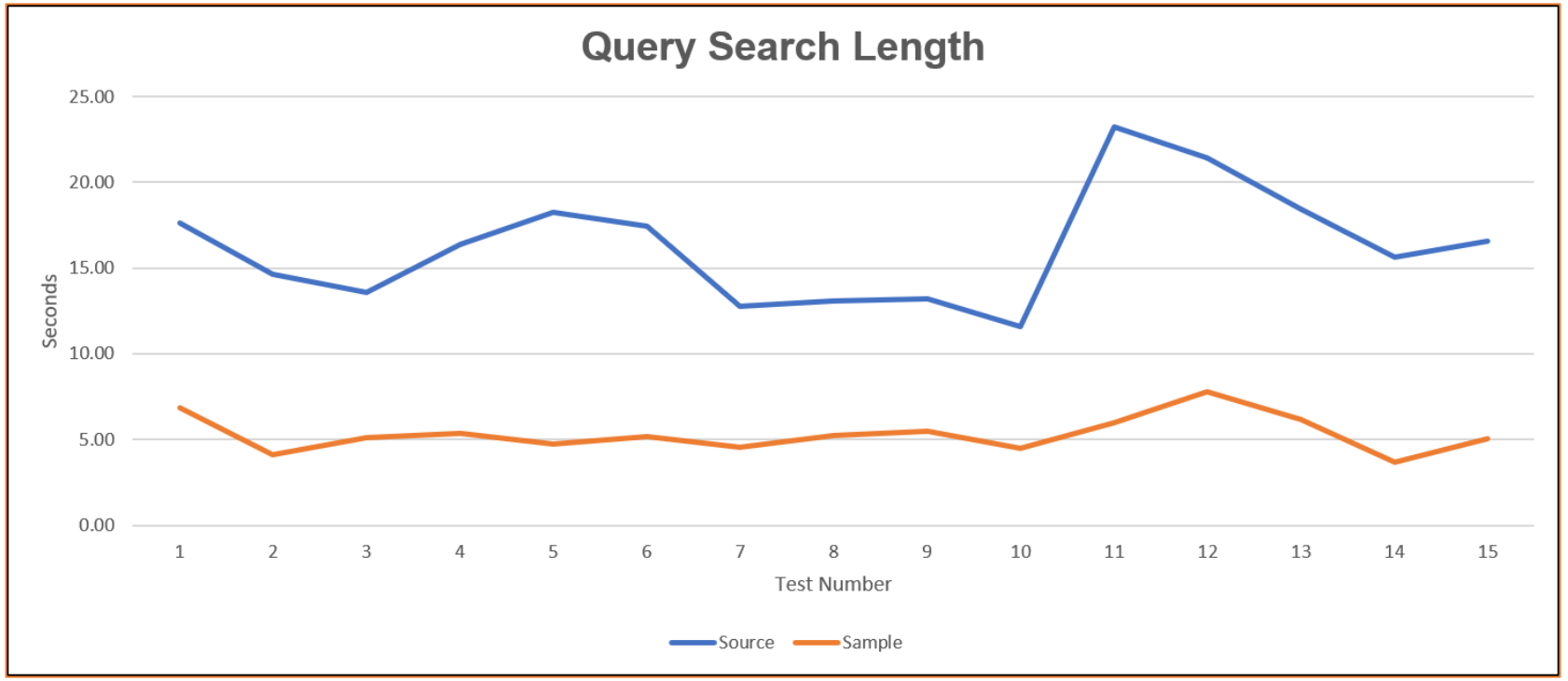
Step 4 – Using the created SRSWOR, and following the join path, the rest of the sample relations are constructed. Overall, a total of fifteen queries are tested five times each, over both the 1GB and 10GB source dataset, as well as the 1GB and 10GB sample dataset..

Results

1GB Dataset

Results

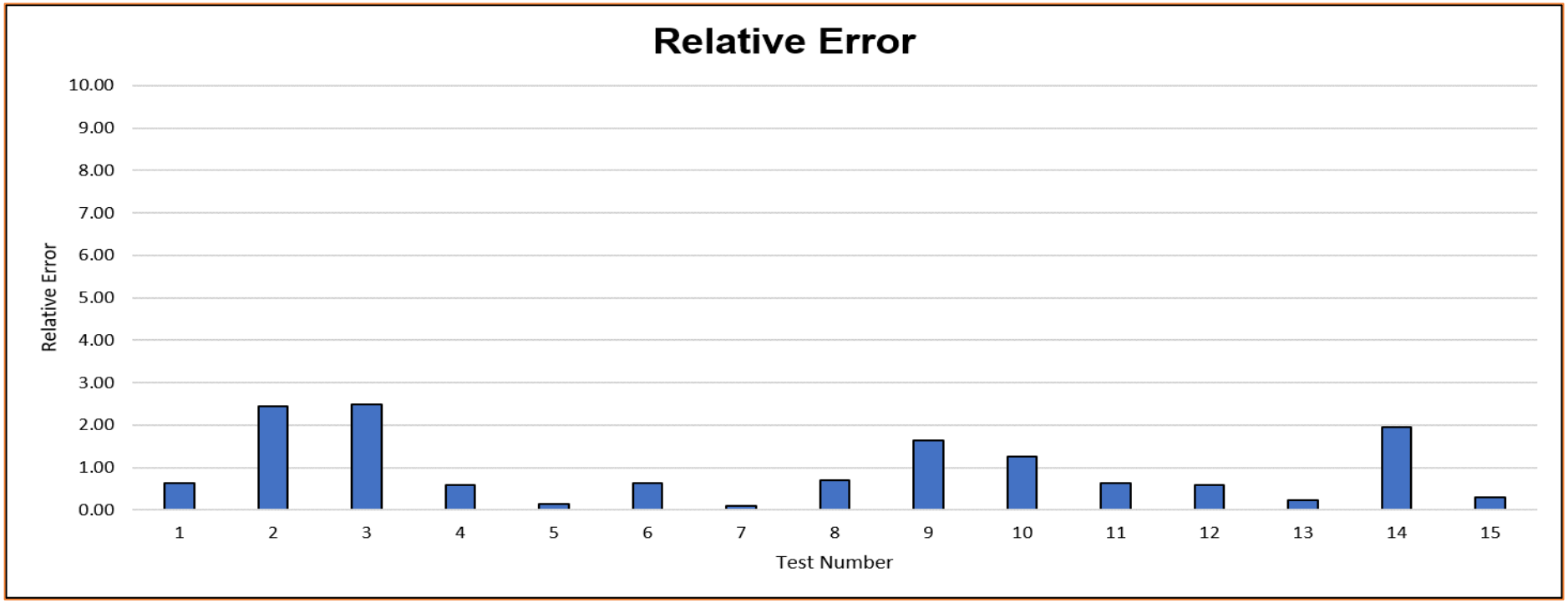
Type	Average	High	Low
Source	16.26	23.24	11.57
Sample	5.33	7.81	3.71



Results

- Average Speed Up from sample, over the source would be 205%
  - Largest Speed Up was 321.71%, lowest was 141.65%

Results



Results

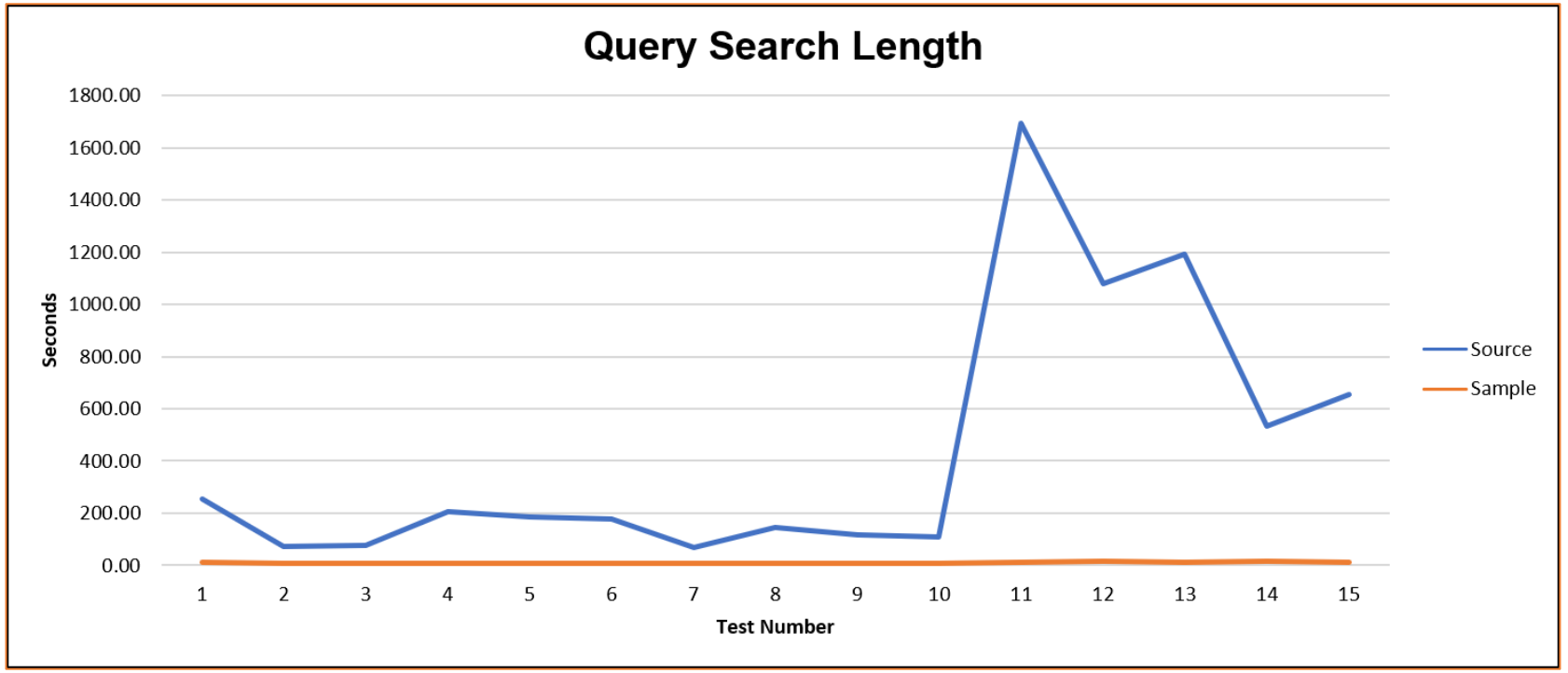
- Average count of tuples source dataset - 2,520,952
- Average count of tuples sample dataset - 25,385
- Average join estimation results - 2,538,533
- Average Relative Error 1GB dataset 0.96%

Results

10GB Dataset

Results

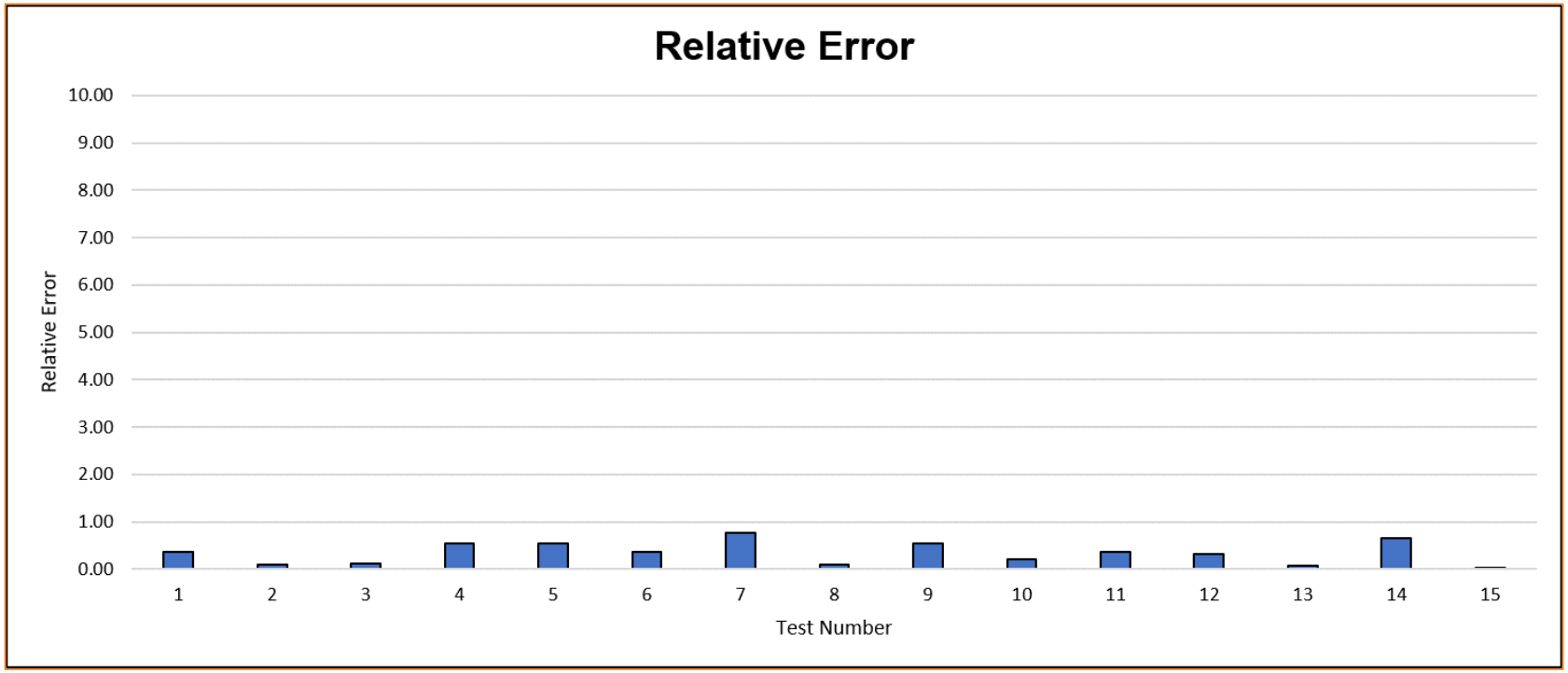
Type	Average	High	Low
Source	437.49	1692.87	69.57
Sample	9.53	14.84	7.00



Results

- Average Speed Up from sample, over the source would be 4489.44%
  - Largest Speed Up was 16,071.9%, lowest was 758.39%

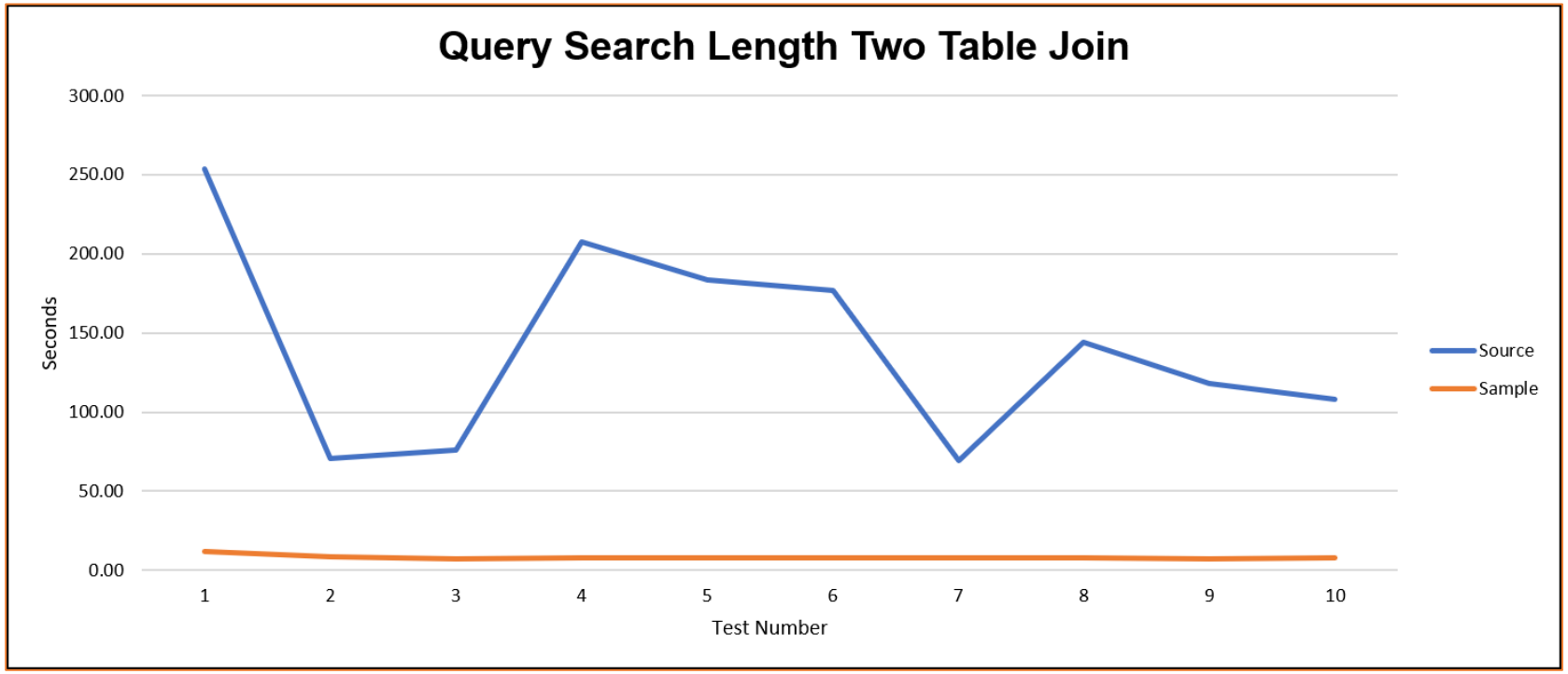
Results



Results

- Average count of tuples source dataset – 25,208,072
- Average count of tuples sample dataset – 251,168
- Average join estimation results – 25,116,820
- Average Relative Error 1GB dataset 0.34%

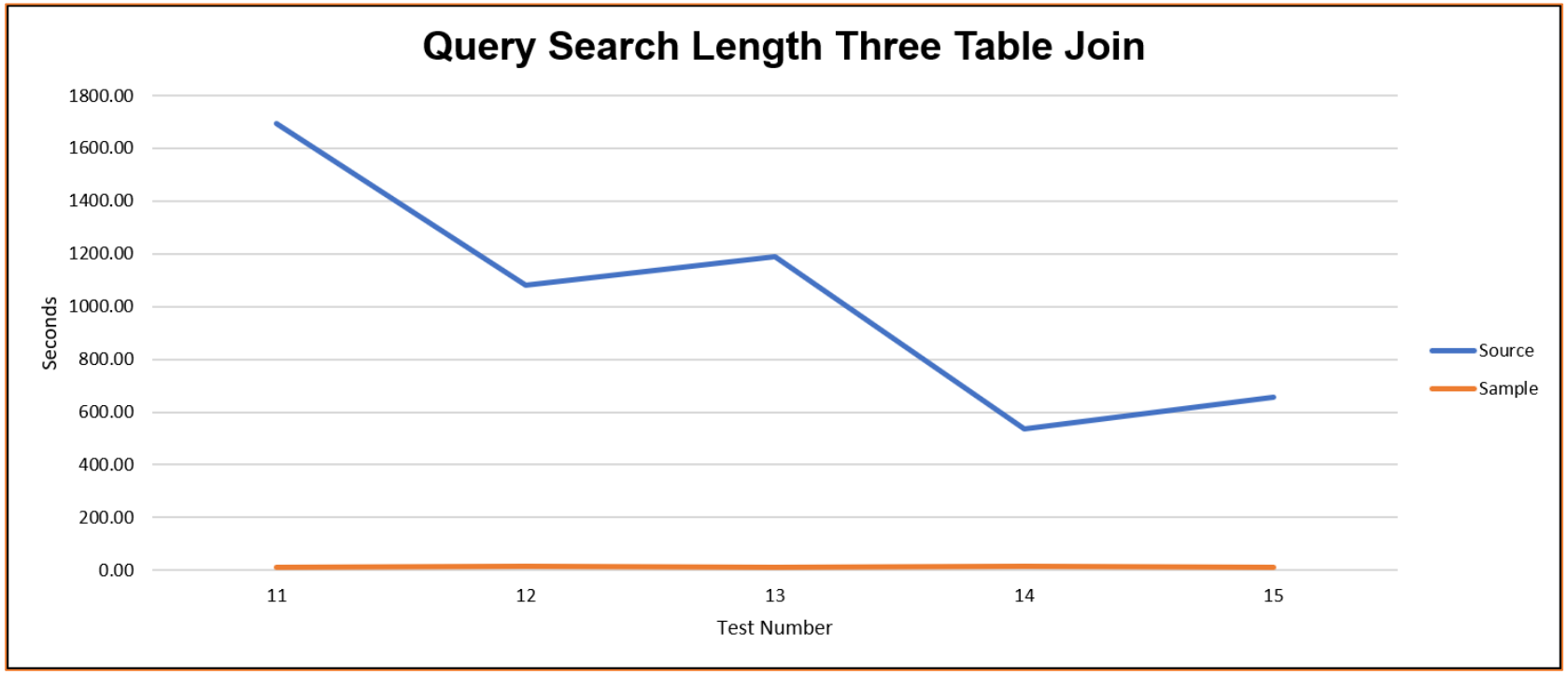
Results



Results

- High Query Source - 253.37
- High Query Sample – 11.70

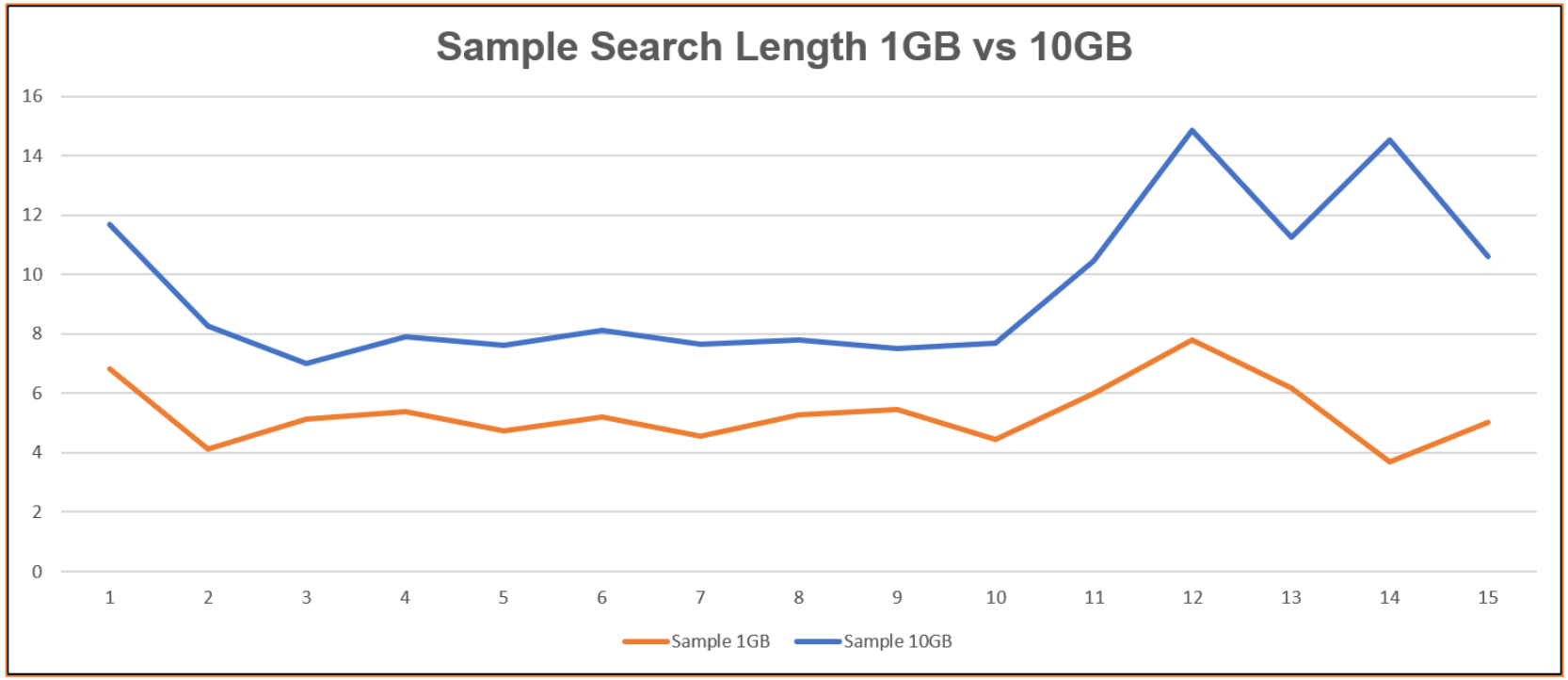
Results



Results

- High Query Source – 1692.87
- High Query Sample – 14.84

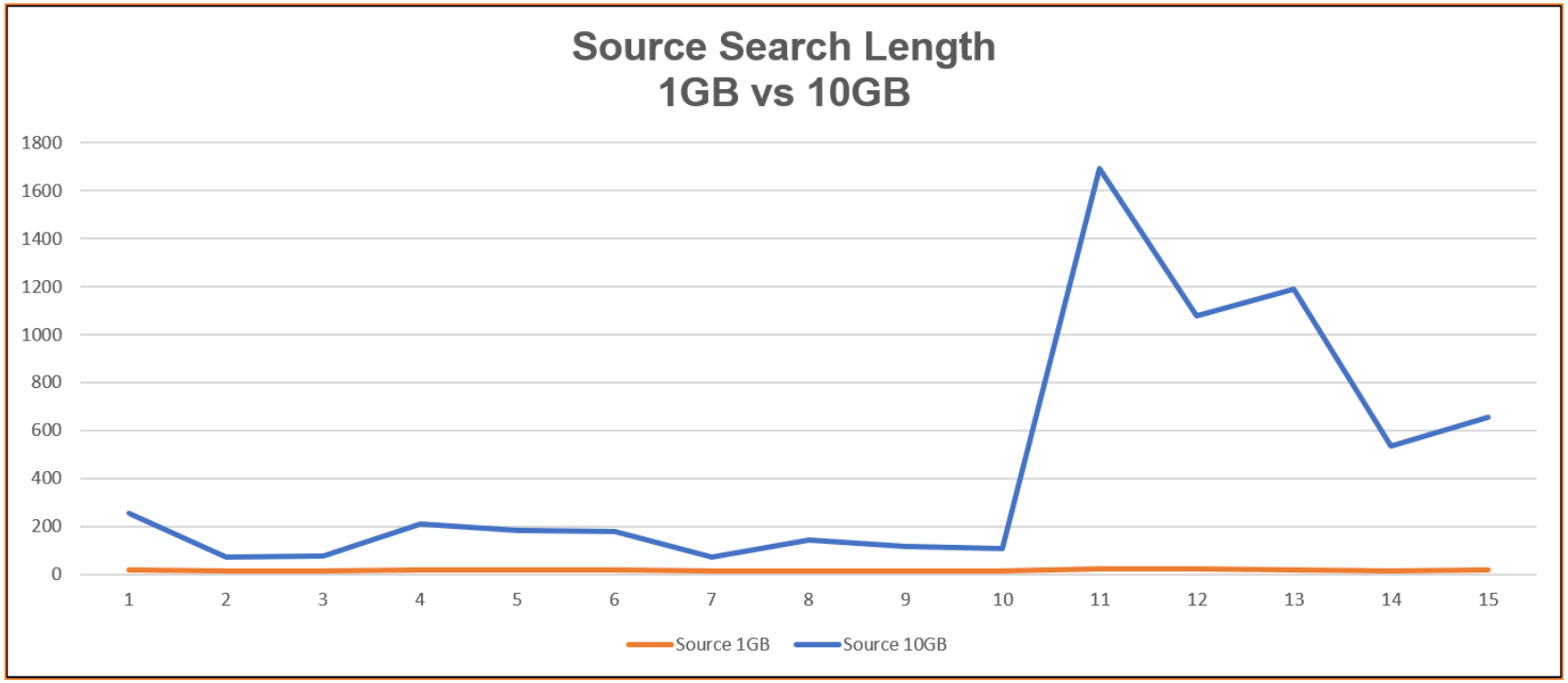
Results



Results

- Sample 1GB High – 7.81
- Sample 10GB High – 14.84

Results



Results

- Source 1GB High - 23.24
- Source 10GB High – 1692.87

Conclusion

Cs2 on Big Datasets was introduced

- CS2 maintained accuracy
- CS2 maintained speed increase

Discovered from research

- CS2 maintained a constant speed throughout testing
  - Continued to produce low search query length, even when source relations ballooned
- CS2 speed increase improved as dataset grew larger

Based off results

- CS2 proved to be successful in query optimization and a more efficient alternative to other optimizers such as Join Synopsis, in regards to storage requirements

Future Works

Source Estimator used in research

- Future research will seek
  - Use No-Source estimator with JR and reverse estimation for join query estimation
  - Use No-Source estimator with JR and reverse estimation to prove CS2 also excels with AQP, ultimately providing the ability to accurately estimate customer trends, among other business analytics, at a fraction of the query processing time.

Citations

[1] B. Marr, “How much data do we create every day? the mind-blowing stats everyone should read,” Jul 2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/05/21/ how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/ #181cc81560ba

[2] D. Laney, “3d data management: Controlling data volume, velocity and variety,” META group research note, vol. 6, no. 70, p. 1, 2001. 35

[3] “The four v’s of big data.” [Online]. Available: https://www.ibmbigdatahub. com/infographic/four-vs-big-data

[4] J. Cano, “The v’s of big data: Velocity, volume, value, variety, and veracity,” Mar 2014. [Online]. Available: https://www.xsnet.com/blog/bid/ 205405/the-v-s-of-big-data-velocity-volume-value-variety-and-veracity

[5] “Number of customer transactions home depot/lowe’s worldwide 2011-2017 | statistic.” [Online]. Available: https://www.statista.com/statistics/318849/ number-of-customer-transactions-at-the-home-depot-and-lowe-s-worldwide/

[6] S. P. Bappalige, “An introduction to apache hadoop for big data.” [Online]. Available: https://opensource.com/life/14/8/intro-apache-hadoop-big-data

[7] A. Shenoy, Hadoop Explained. Packt Publishing Ltd, 2014.

[8] “Hadoop vs hive - find out the best differences,” Oct 2018. [Online]. Available: https://www.educba.com/hadoop-vs-hive/

[9] F. Yu, W.-C. Hou, C. Luo, D. Che, and M. Zhu, “Cs2: A new database synopsis for query estimation,” in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’13. New York, NY, USA: ACM, 2013, pp. 469–480. [Online]. Available: http://doi.acm.org/10.1145/2463676.2463701

[10] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, “Join synopses for approximate query answering,” SIGMOD Rec., vol. 28, no. 2, pp. 275–286, Jun. 1999. [Online]. Available: http://doi.acm.org/10.1145/304181.304207

Thank you!

Any questions or suggestions?