Advanced Data Management

Medha Atre

Office: KD-219

atrem@cse.iitk.ac.in

July 28, 2016

Goals

- Learn tools and techniques of the "big data" management.
- Big data encompasses relational data, graph data, text data, and any other data (media etc).
- "Graph shaped" data has become predominant as it captures the ideology of *internet* and *social networks* – a network of information, persons, objects, and things in general.
- E.g., DBPedia[1] is a "graphical" representation of the Wikipedia network. UniProt[2] is a graphical representation proteins and genes network, MusicBrainz[3] is a graphical representation of music database.

---

1 http://wiki.dbpedia.org/

2 http://www.uniprot.org/

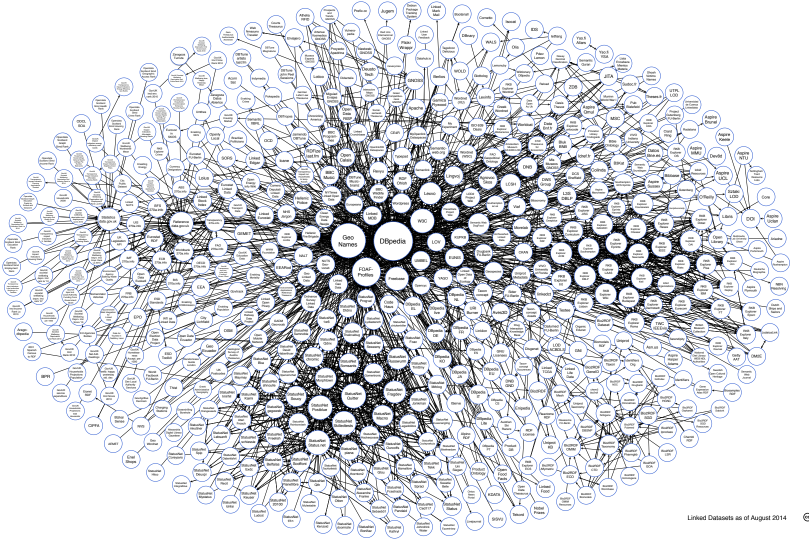
3 https://musicbrainz.org/

Goals

- Is this graph data of any significance?
  - DBPedia (Wikipedia network in RDF format, pivotal in the success of IBM Watson supercomputer in the *Jeopardy* challenge, along with the Yago RDF network)[4]
  - Adaption of RDFa by Google, and RDF by Facebook.
- Understand the challenges of management of this data from the "scalability" point of view – how would you manage such a huge data? Would you use
  - one commodity computer, or
  - a cluster of commodity computers, or
  - a supercomputer?

[4] http://en.wikipedia.org/wiki/Watson_(computer)#Data

A glimpse of the Linked Data network



Linking Open Data cloud diagram 2014, by M. Schmachtenberg, C. Bizer, A. Jentzsch, R. Cyganiak http://lod-cloud.net/

Course Structure

- 50% – Course project, supposed to be innovative, chosen from the cutting-edge research topic. A list of research papers of interesting topics will be referred to by the instructor. Students may choose an outside topic of their interest as long as it is relevant to the broader theme of the course. Breakup of the grades:
  - 5% – A written proposal with a literature review.
  - 30% – Implementation and execution on the Linux environment.
  - 15% – Demonstrable evaluation results.
- All the course projects need to be well defined and need instructor's approval by the middle of the semester.

Course Structure

- Course projects can be done in the groups of 1, 2, or 3 students together.
  - Groups of size more than 3 need special approval. Such a group needs to demonstrate a significant component of the project by the middle of the semester to get the approval.
- Recommended size of a group is 2 or 3.
- 15% – Bi-weekly (once in 2 weeks) assignments.
- 15% – Mid-semester State-of-the-art paper presentation.
- 20% – Final detailed course project report (expected to be written as professionally as possible like a research paper).

Office hours: Mon-Thur 12-1pm (else by prior appointment).
Course webpage: www.cse.iitk.ac.in/users/atrem/courses/cs698f2016fall/

Academic Integrity

We value academic integrity and honesty above good grades.

Every student is required to go through the CSE department academic integrity (anti-cheating) policy given on http://www.cse.iitk.ac.in/pages/AntiCheatingPolicy.html

After the add/drop deadline, each registered student will be required to sign an acknowledgement of having read this policy.

There will be 0 tolerance shown towards plagiarism, copying[5], and any other dishonest means.

[5] Using open source codes or components of them with proper acknowledgement is allowed.

Relational Data

STORE

Store_key	City	Region
1	New York	**East**
2	Chicago	Central
3	Atlanta	**East**
4	Los Angeles	**West**
5	San Francisco	**West**
6	Philadelphia	**East**
.	.	.
.	.	.

PRODUCT

Product_key	Description	Brand
1	Beautiful Girls	MKF Studios
2	Toy Story	**Wolf**
3	Sense and Sensibility	Parabuster Inc.
4	Holiday of the Year	**Wolf**
5	Pulp Fiction	MKF Studios
6	The Juror	MKF Studios
7	From Dusk Till Dawn	Parabuster Inc.
8	Hellraiser: Bloodline	Big Studios
.	.	.
.	.	.

SALES_FACT

Store_key	Product_key	Sales	Cost	Profit
1	6	2.39	1.15	1.24
1	2	16.7	6.91	9.79
2	7	7.16	2.75	4.40
3	2	4.77	1.84	2.93
5	3	11.93	4.59	7.34
5	1	14.31	5.51	8.80
.	.	.	.	.
.	.	.	.	.

Figure taken from https://docs.oracle.com/.

SQL Queries

Find description and brand of all the products in all the stores in the New York city.

SELECT PRODUCT.Description, PRODUCT.Brand
WHERE STORE.City="New York" AND
STORE.Store_key=SALES_FACT.Store_key AND
SALES_FACT.Product_key=PRODUCT.Product_key

Query Planning and Optimization

- Cost based estimation for selecting the best query plan.
- Costs are in terms of number of rows (disk blocks) you have to access to perform a join operation, whether the table has any indexes on it etc.

In a nested loop join, suppose STORE has S pages and t_S tuples per page, and SALES_FACT has F pages and t_F tuples per page, then with STORE as the *outer* relation and SALES_FACT as the *inner* relation, the cost of the join between the two will be: $S + t_S * S * F$.

In a block nested loop join with B buffer pages, the cost of this join will be: $S + F * \lceil \frac{S}{B-2} \rceil$ where 2 buffer pages are reserved for the inner relation SALES_FACT.

Query Optimization – well known techniques

- Create indexes on frequently joined columns.
- Hash-joins.
- Pipelining of data through the operators to avoid creation of temporary intermediate relation – popular method in the contemporary database systems.
- Creation of *join-indices* for frequently used joins.
- Pre-join pruning of data based on the schema of the data or other techniques such as *semi-joins*.
- *Column-wise* storage instead of traditional *row-wise* storage.
- Data compression – column compression, compressed bit-vectors, delta encoding techniques.

Challenges

- When the difference between the size of the disk resident data and available main memory becomes larger.
- When the data cannot be stored on the single disk on a single computer.
- When cost-estimation methods of query planning are rendered ineffective due to heavy skew in the data – common in the "graph shaped data".

Challenges

- E.g., A graph with 1 *billion* edges (10^9) cannot be fully loaded in memory even on a machine with 32GB main memory. Its disk resident size in the uncompressed form is about 400GB, e.g., UniProt network.
- Cost of a PC with 8GB RAM, 1TB disk Rs. 30,000.
- Cost of a PC with 32GB RAM 1TB disk is Rs. 1,75,280.

If your data processing requires cumulative X amount of memory and Y amount of disk space:

$$(cost(PC(X, Y)) \times 1) \gg (cost(PC(\frac{X}{n}, \frac{Y}{n})) \times n)$$

Where n is the number of commodity compute nodes. That is what Google is doing through their *server farms*!

Challenges

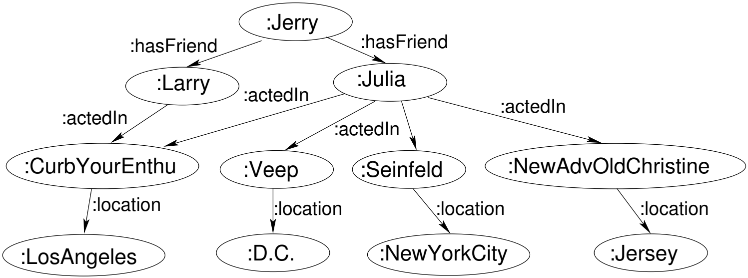
- When the data gets distributed on a farm of servers, instead of a single server, the cost computation changes.
- Now network communication cost has to be included while planning a query.

Let us go back to our example of the join between STORE and SALES_FACT tables. Assuming none of the join data is available locally, and STORAGE table has to be communicated across the entire cluster of 5 compute nodes, an approximate cost computation of the join will be:

$$5 * (\frac{S}{5} + 4 * 2 * Comm(\frac{S}{5}) + t_S * S * \frac{F}{5})$$

We can do better than this... and in this course we will see how!

Graph data of TV sitcoms



Graph data and queries

Data

:Jerry	:hasFriend	:Larry
:Jerry	:hasFriend	:Julia
:Larry	:actedIn	:CurbYourEnthu
:Julia	:actedIn	:Seinfeld
:Julia	:actedIn	:Veep
:Julia	:actedIn	:CurbYourEnthu
:Julia	:actedIn	:NewAdvOldChristine
:Seinfeld	:location	:NewYorkCity
:Veep	:location	:D.C.
:CurbYourEnthu	:location	:LosAngeles
:NewAdvOldChristine	:location	:Jersey

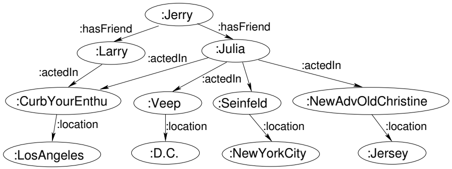
SPARQL

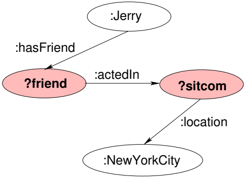
SELECT ?friend ?sitcom **WHERE** {
:Jerry :hasFriend **?friend** .
?friend :actedIn **?sitcom** .
?sitcom :location :NewYorkCity .
}

Eqv. SQL query

SELECT t1.o, t2.o from rdf as t1, rdf as t2, rdf as t3 **WHERE** t1.s=":Jerry" and t1.p=":hasFriend" and t2.p=":actedIn" and t3.p=":location" and t3.o=":NewYorkCity" and **t1.o=t2.s and t2.o=t3.s**

Graphical Representation





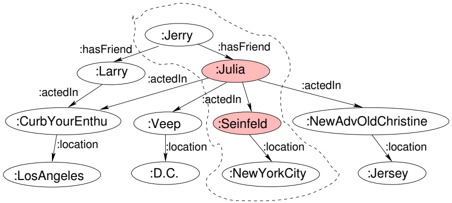
Query Evaluation Results

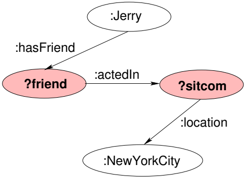
Data

:Jerry	:hasFriend	:Larry
:Jerry	:hasFriend	:Julia
:Larry	:actedIn	:CurbYourEnthu
:Julia	:actedIn	:Seinfeld
:Julia	:actedIn	:Veep
:Julia	:actedIn	:CurbYourEnthu
:Julia	:actedIn	:NewAdvOldChristine
:Seinfeld	:location	:NewYorkCity
:Veep	:location	:D.C.
:CurbYourEnthu	:location	:LosAngeles
:NewAdvOldChristine	:location	:Jersey

Query

SELECT ?friend ?sitcom **WHERE** {
:Jerry :hasFriend **?friend** .
?friend :actedIn **?sitcom** .
?sitcom :location :NewYorkCity .
}





Graphs – Types of problems

- Pattern matching, a.k.a., joins.
- Optional pattern matching, where part of the pattern is match is optional, a.k.a., left-outer-joins.
- Path pattern queries, *hard problems* –
  - Given a regular expression, find all pairs of nodes that have at least one path matching that expression – recursive join queries.
  - Given a regular expression, and a pair of nodes, find all paths that satisfy that expression.
- Reachability queries – static and dynamic graphs.
- Keyword search over graphs.

Graphs – Challenges

- Lacks schema, hence standard techniques of *normalization* as in relational DBs not applicable.
- Has a heavy skew in the data – a few nodes with very high degree, and a large number of nodes with very less degree.
- Lots of *strongly connected components* and hence cycles.
- Distribution strategies need to take care of graph structure, to avoid *large cuts* – methods like METIS[6] help in efficient graph partitioning.

[6] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

Solution space – general purpose

- Apache Hadoop – Map Reduce framework.
- Apache SPARK – in-memory Hadoop like framework.
- H20 – Big data analysis.
- Apache Mahout – for substrate independent big scale machine learning tasks, typically run on Hadoop, SPARK, H20, Flink.

Solution space – relational data

- Single compute node:
  - Row-wise stores – MySQL, PostgreSQL, Many other commercial stores like IBM DB2.
  - Column-wise stores – MonetDB, Virtuoso.
- Cluster computing:
  - Cloudera, Cassandra, BigTable, HBase, HyperTable (inspired by BigTable).

Solution space – graph data

- Single compute node:
  - BitMat[7] [Atre2008, Atre2010, Atre2015, Atre2016] – for pattern matching, a.k.a. joins.
  - RDF-3X [Neumann2010]
  - Neo4j, HypergraphDB.
- Cluster computing:
  - Pregel by Google (Apache Giraph as opensource version)
  - Trinity by Microsoft, GraphX library (in Apache SPARK)
  - H-RDF-3X [Huang2011], SHARD [Rohloff2011], TriAD [Gurajada2014].

7 http://bitmatrdf.sourceforge.net

Flow of the course

- Go over graph data processing systems on a single node compute environment, to understand the problems and solutions in detail.
- Through that understand the scalability problems.
- Gradually move towards distributed systems, and understand the solutions for scalability.
- Through the above exercises, identify problems in the space of both – scalability of the existing solutions and problems that lack good solutions even on the single node compute environment.
- Introduction to open and/or advanced problems over graph and relational data processing for further research.