Alternate *ACM* SIG Proceedings Paper in LaTeX Format*

[Extended Abstract] †

Johannes Spießberger Ralph Hoch Carola Gabriel

ABSTRACT

Targeted advertising is one of the key revenue sources for internet services. While traditional approaches tried to suggest ads to users based on statistics derived from historical data, modern approaches try to make use of big data. This paper tries to give a short overview of current scientific trends in NoSQL and big data management. As graph databases are especially fitting for modelling social interactions, this paper puts an emphasis on this type of NoSql.

General Terms

Big Data, targeted advertising, NoSQL, graph database

Keywords

Big Data, targeted advertising, NoSQL, graph database

1. INTRODUCTION

The recent years have shown a massive growth in data used for analysis in a broad range of fields. Big data as a concept for handling this amount of information has become a huge field for scientific research and business models alike. One of the cornerstones of the technical implementation of such systems is the usage of NoSQL databases. While these are available in many different flavours, graph based approaches are the one that differ the most from traditional database systems. Modelling social interactions as graphs and extracting various information is, among other applications, one of todays key techniques for targeted advertising. TODO cite http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=1579567 While the usage of such graph databases makes for a natural abstraction of some real life observations, the technical implementations of the graph database itself becomes more challenging than those of traditional RDBMs system. The following sections will summarize some recent research concerning the inner workings of graph databases.

2. GPU BASED FREQUENT GRAPH MINING

With the availability of CUDA and opencl GPUs have become a source of cheap computation power for significantlly less money than general purpose CPUs. While not applicable to all types of computational loads, there are various fields which benefit immensly from a high parallelization degree TODO source. Especially loads without the need of communication between the threads are well suited for GPU architectures TODO source. The following sections will describe some applications in graph databases.

Source: http://jmlr.org/proceedings/papers/v36/kessl14.pdf Frequent graph mining describes the search of reoccuring sub patterns in a graph. Among various applications this technique can be used to find similar communities within social networks TODO source. TODO bild subgraph As this problem has shown to be NP-complete TODO source it is especially important to find scalable algorithms to deal with this problem. The GPU-based graph mining algorithm combines concepts from gSpan TODO source and DMTL TODO source. In order to achieve good performance it is necessary to change the data structure of the graph in main memory. Hashmaps or linked lists are not well suited for GPUs as data is not cached and therefore access locality becomes a performance consideration. For various benchmarks the usage of GPUs has lead to a significant speedup. TODO Benchmark bild This work can also be extended to different graph algorithms like closed graph, maximal graph and temporal graph mining.

3. DIFFERENTIAL QUERIES

http://ceur-ws.org/Vol-1133/paper-34.pdf http://download.springe the rigid predefined schema of traditional database systems is one of the key features of NoSQL. While offering more flexibility in terms of development, users have a hard time gaining deep knowledge of the data and its structure. This mostly manifests in queries giving unexpected or empty result sets. A query in a graph database can be understood as a pattern that has to match parts of the graph. To help the user understand why he does not get the expected result subgraphs between the query and the dataset are calculated, so it can be shown which part of the query leads to an empty

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

†A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2ε and BibTeX* at `www.acm.org/eaddress.htm`

WOODSTOCK '97 El Paso, Texas USA

result set. Using algorithms to detect maximum common connected subgraphs, nodes in the query can be partitioned into discovered and undiscovered conditions. Due to the nature of these operations are large number of intermediate results may lead to performance problems. This problem may be mitigated by introducing Top-K differential queries. Such queries are still modeled as graphs but have additional properties allowing for ranking the intermediate results by relevance. These weights are given by user and can be either maximized or minimized.

4. SLQ

Another approach of easing the writing of graph database queries for non professional users is SLQ language. SLQ interprets queries based on keywords which are then further mapped to onthologies. While a strict replacing of words would lead to high number of queries, the mapping of keywords is based on their semantical closenes. To get the most relevant results the generated queries and their transformations are weighted. By using used feedback and query logs SLQ is able to learn how to apply a reasonable weighting. To create a SLQ query the user specifies it by drawing a graph and attaching the keywords. TODO BILD

5. PARALLEL ADJACENCY LISTS

http://arxiv.org/pdf/1403.0701.pdf Modelling social networks like Twitter has some particular challenges for graph DB which come from the uneven distribution of edges between nodes. Such graphs have the tendency to have a small number of nodes with a high number of edges and a high number of nodes with a small number of edges. TODO zitat aus Paper 11 This makes partitioning a graph to allow for localized acces hard. Partitioned Adjacecny Lists PAL tries to reduce the number of random accesses while minimizing the storage space required. Edges are stored as pairs of vertices and are partitioned by a range of destination vertices and are ordered by the source ID. These partitiones must not be of equal length but should be chosen such that a partition can fit into memory. This theoretically limits the number of incoming edges a vertex may have. As the graph connectivity is fully represented by the edge partitions, vertex data can be stored seperately and is partitioned based on the edge partitions. Retrieving verted data can be achieved by simly calculating an offset from the edge partitions. Edge partitions are immutable data structures and do not allow direct insertion of edges. To counter this problem new edges a stored in a buffer and inserted if their number exceeds a certain threshold. During search operations these buffers are also considered. Todo zahlen und benchmark bilder

6. DISTRIBUTED GRAPH DATABASE

Current graph databases are mainly inteded to run on a single instance and therefore face performance and availability problems. TODO Zitat aus paper Acadia is a distributed graph database intented to run in hybrid cloud settings. By utilizing graph partitioning and a Master-Worker pattern it is well fit to operate in an environment with less than 100 workers. Each distributed process has an associated local data store with an assigned storage quota. When adding a new graph the partitioning happens via Hadoop and Metis and the subgraphs are then inserted into local Neo4j instances.

7. CONCLUSIONS

TODO