BIG DATA INTEGRATION
RESEARCH AT THE UNIVERSITY OF LEIPZIG

ERHARD RAHM, UNIV. LEIPZIG

www.scads.de
Founded in 1409

Now about 30,000 students in 14 faculties

Computer science
  - 13 professorships and 2 junior professors
  - 150 PhD students and postdocs (120 by third party funding)
Two Centers of Excellence for Big Data in Germany

- ScaDS Dresden/Leipzig
- Berlin Big Data Center (BBDC)

ScaDS Dresden/Leipzig (Competence Center for Scalable Data Services and Solutions Dresden/Leipzig)

- scientific coordinators: Nagel (TUD), Rahm (UL)
- start: Oct. 2014
- duration: 4 years (option for 3 more years)
- initial funding: ca. 5.6 Mio. Euro
- Bundling and advancement of existing expertise on Big Data
- Development of Big Data Services and Solutions
- Big Data Innovations
Avantgarde-Labs GmbH
Data Virtuality GmbH
E-Commerce Genossenschaft e. G.
European Centre for Emerging Materials and Processes Dresden
Fraunhofer-Institut für Verkehrs- und Infrastruktursysteme
Fraunhofer-Institut für Werkstoff- und Strahltechnik
GISA GmbH
Helmholtz-Zentrum Dresden - Rossendorf

Hochschule für Telekommunikation Leipzig
Institut für Angewandte Informatik e. V.
Landesamt für Umwelt, Landwirtschaft und Geologie
Netzwerk Logistik Leipzig-Halle e. V.
Sächsische Landesbibliothek – Staats- und Universitätsbibliothek Dresden
Scionics Computer Innovation GmbH
Technische Universität Chemnitz
Universitätsklinikum Carl Gustav Carus
STRUCTURE OF THE CENTER

Big Data Life Cycle Management and Workflows

- Data Quality / Data Integration
- Knowledge Extraktion
- Visual Analytics

Efficient Big Data Architectures

Service center

- Life sciences
- Material and Engineering sciences
- Environmental / Geo sciences
- Digital Humanities
- Business Data
- Data-intensive computing  W.E. Nagel
- Data quality / Data integration  E. Rahm
- Databases  W. Lehner, E. Rahm
- Knowledge extraction/Data mining  C. Rother, P. Stadler, G. Heyer
- Visualization  S. Gumhold, G. Scheuermann
- Service Engineering, Infrastructure  K.-P. Fähnrich, W.E. Nagel, M. Bogdan
APPLICATION COORDINATORS

- Life sciences  G. Myers
- Material / Engineering sciences  M. Gude
- Environmental / Geo sciences  J. Schanze
- Digital Humanities  G. Heyer
- Business Data  B. Franczyk
AGENDA

- ScaDS Dresden/Leipzig

- Big Data Integration
  - Introduction
  - Matching product offers from web shops
  - DeDoop: Deduplication with Hadoop

- Privacy-preserving record linkage with PP-Join
  - Cryptographic bloom filters
  - Privacy-Preserving PP-Join (P4Join)
  - GPU-based implementation

- Big Graph Data
  - Graph-based Business Intelligence with BIIIG
  - GraDoop: Hadoop-based data management and analysis

- Summary and outlook
BIG DATA ANALYSIS PIPELINE

Data acquisition → Data extraction/cleaning → Data integration/annotation → Data analysis and visualization → Interpretation

Heterogeneity, Volume, Velocity, Privacy, Human collaboration
Thousands of data sources (shops/merchants)

Millions of products and product offers

Continuous changes

Many similar, but different products

Low data quality
LEARNING-BASED MATCH APPROACH

1. Pre-processing
   - Product Code Extraction
   - Manufacturer Cleaning
   - Automatic Classification

2. Training
   - Training Data Selection
   - Matcher Application
   - Classifier Learning

3. Application
   - Blocking (Manufacturer + Category)
   - Matcher Application
   - Classification

Product Match Result

Product Offers
HOW TO SPEED UP OBJECT MATCHING?

- **Blocking** to reduce search space
  - group similar objects within blocks based on *blocking key*
  - restrict object matching to objects from the same block

- **Parallelization**
  - split match computation in sub-tasks to be executed in parallel
  - exploitation of Big Data infrastructures such as Hadoop (Map/Reduce or variations)
GENERAL OBJECT MATCHING WORKFLOW

Map Phase: Blocking

Reduce Phase: Matching
Parallel execution of data integration/match workflows with Hadoop

Powerful library of match and blocking techniques

Learning-based configuration

GUI-based workflow specification

Automatic generation and execution of Map/Reduce jobs on different clusters

Automatic load balancing for optimal scalability

Iterative computation of transitive closure (extension of MR-CC)

“This tool by far shows the most mature use of MapReduce for data deduplication”

www.hadoopsphere.com
DEDoop Overview

General ER workflow

Machine Learning

Blocking

Similarity Computation

Match Classification

Classifier Training Job

Data Analysis Job

Blocking-based Matching Job

Dedoop’s general MapReduce workflow

Core

- Decision Tree
- Logistic Regression
- SVM
- ...

- Standard Blocking
- Sorted Neighborhood
- PPJoin+
- Blocking Key Generators
  - Prefix
  - Token-based
- ...

- Edit Distance
- n-gram
- TFIDF
- ...

- Threshold
- Match rules
- ML model
- ...

$T \subseteq R \times S \times [0,1]$
ScaDS Dresden/Leipzig

Big Data Integration
- Introduction
- Matching product offers from web shops
- DeDoop: Deduplication with Hadoop

Privacy-preserving record linkage with PP-Join
- Cryptographic bloom filters
- Privacy-Preserving PP-Join (P4Join)
- GPU-based implementation

Big Graph Data
- Graph-based Business Intelligence with BIIIG
- GraDoop: Hadoop-based data management and analysis

Summary and outlook
Need for comprehensive privacy support ("privacy by design")
- Privacy-preserving publishing of datasets
- Privacy-preserving record linkage
- Privacy-preserving data mining

Privacy-preserving record linkage
- object matching with encrypted data to preserve privacy
- conflicting requirements: high privacy, scalability and match effectiveness
- use of central linking unit (Trusted third party) vs. symmetric approaches (Secure Multiparty Computing)
- effective and simple encryption uses cryptographic bloom filters (Schnell et al, 2009)

- tokenize all match-relevant attribute values, e.g. using bigrams or trigrams
  - typical attributes: first name, last name (at birth), sex, date of birth, country of birth, place of birth

- map each token with a family of one-way hash functions to fixed-size bit vector (fingerprint)
  - original data cannot be reconstructed

- match of bit vectors (Jaccard similarity) is good approximation of true match result
Sim\textsubscript{Jaccard} (r\textsubscript{1}, r\textsubscript{2}) = (r\textsubscript{1} \land r\textsubscript{2}) / (r\textsubscript{1} \lor r\textsubscript{2})

Sim\textsubscript{Jaccard} (r\textsubscript{1}, r\textsubscript{2}) = 7/11
one of the most efficient similarity join algorithms
- determine all pairs of records with $\text{sim}_{\text{Jaccard}}(x,y) \geq t$

use of filter techniques to reduce search space
- length, prefix, and position filter

relatively easy to run in parallel

good candidate to improve scalability for PPRL

evaluate set bit positions instead of (string) tokens
- matching records pairs must have similar lengths

\[ \text{Sim}_{\text{Jaccard}}(x, y) \geq t \Rightarrow |x| \geq |y| \ast t \]

- length / cardinality: number of set bits in bit vector

- Example for minimal similarity \( t = 0.8 \):

<table>
<thead>
<tr>
<th>ID</th>
<th>Bit vector</th>
<th>card.</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1 0 1 0 0 0 0 0 1 1 0 0 0</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>0 0 0 1 1 1 1 1 1 1 0 0 0</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>0 1 0 1 1 1 1 1 1 1 1 0 0 0</td>
<td>8</td>
</tr>
</tbody>
</table>

- record B of length 4 cannot match with C and all records with greater length (number of set positions), e.g., A

length filter \( 7 \ast 0.8 = 5.6 > 4 \)
Similar records must have a minimal overlap $\alpha$ in their sets of tokens (or set bit positions)

$$\text{Sim}_{\text{Jaccard}}(x, y) \geq t \iff \text{Overlap}(x, y) \geq \alpha = \left\lceil \left( \frac{t}{1+t} \cdot (|x| + |y|) \right) \right\rceil$$

Prefix filter approximates this test

- reorder bit positions for all fingerprints according to their overall frequency from infrequent to frequent
- exclude pairs of records without any overlap in their prefixes with

$$\text{prefix\_length}(x) = \left\lceil \left( (1-t) \cdot |x| \right) + 1 \right\rceil$$

Example ($t = 0.8$)

<table>
<thead>
<tr>
<th>ID</th>
<th>reordered fingerprint</th>
<th>card.</th>
<th>prefix fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1 0 1 0 0 0 0 0 1 1 0 0 0 0 4</td>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0 0 0 1 1 1 1 1 1 1 0 0 0 0 7</td>
<td>0 0 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0 1 0 1 1 1 1 1 1 1 0 0 0 0 8</td>
<td>0 1 0 1 1</td>
<td></td>
</tr>
</tbody>
</table>

AND operation on prefixes shows non-zero result for C and A so that these records still need to be considered for matching
P4JOIN: POSITION FILTER

- improvement of prefix filter to avoid matches even for overlapping prefixes
  - estimate maximally possible overlap and checking whether it is below the minimal overlap $\alpha$ to meet threshold $t$
  - original position filter considers the position of the last common prefix token

- revised position filter
  - record $x$, prefix 1 1 0 1 length 9
  - record $y$, prefix 1 1 1 length 8
  - highest prefix position (here fourth pos. in $x$) limits possible overlap with other record: the third position in $y$ prefix cannot have an overlap with $x$
  - maximal possible overlap = #shared prefix tokens (2) + min (9-3, 8-3) = 7
  - $< \text{ minimal overlap } \alpha = 8$
comparison between NestedLoop, P4Join, MultiBitTree

- MultiBitTree: best filter approach in previous work by Schnell
  - applies length filter and organizes fingerprints within a binary tree so that fingerprints with the same set bits are grouped within sub-trees
  - can be used to filter out many fingerprints from comparison

two input datasets R, S

- determined with FEBRL data generator
  \[ N = [100.000, 200.000, \ldots, 500.000] \]
  \[ |R| = \frac{1}{5} \cdot N, \quad |S| = \frac{4}{5} \cdot N \]
- bit vector length: 1000
- similarity threshold 0.8
- runtime in minutes on standard PC

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dataset size N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100.000</td>
</tr>
<tr>
<td>NestedLoop</td>
<td>6,10</td>
</tr>
<tr>
<td>MultiBitTree</td>
<td>4,68</td>
</tr>
<tr>
<td>P4 Length filter only</td>
<td>3,38</td>
</tr>
<tr>
<td>P4 Length+Prefix</td>
<td>3,77</td>
</tr>
<tr>
<td>P4 Length+Prefix+Position</td>
<td>2,25</td>
</tr>
</tbody>
</table>

- similar results for P4Join and Multibit Tree
- relatively small improvements compared to NestedLoop
Operations on bit vectors easy to compute on GPUs

- Length and prefix filters
- Jaccard similarity

Frameworks CUDA und OpenCL support data-parallel execution of general computations on GPUs

- Program („kernel“) written in C dialect
- Limited to base data types (float, long, int, short, arrays)
- No dynamic memory allocation (programmer controls memory management)
- Important to minimize data transfer between main memory and GPU memory
- partition inputs R and S (fingerprints sorted by length) into equally-sized partitions that fit into GPU memory
  - generate match tasks per pair of partition
  - only transfer to GPU if length intervals per partition meet length filter
  - optional use of CPU thread to additionally match on CPU
GPU-BASED EVALUATION RESULTS

GeForce GT 610
- 48 Cuda Cores@810MHz
- 1GB
- 35€

GeForce GT 540M
- 96 Cuda Cores@672MHz
- 1GB

<table>
<thead>
<tr>
<th></th>
<th>100.000</th>
<th>200.000</th>
<th>300.000</th>
<th>400.000</th>
<th>500.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeForce GT 610</td>
<td>0,33</td>
<td>1,32</td>
<td>2,95</td>
<td>5,23</td>
<td>8,15</td>
</tr>
<tr>
<td>GeForce GT 540M</td>
<td>0,28</td>
<td>1,08</td>
<td>2,41</td>
<td>4,28</td>
<td>6,67</td>
</tr>
</tbody>
</table>

- improvements by up to a factor of 20, despite low-profile graphic cards
- still non-linear increase in execution time with growing data volume
AGENDA

- ScaDS Dresden/Leipzig
- Big Data Integration
  - Introduction
  - Matching product offers from web shops
  - DeDoop: Deduplication with Hadoop
- Privacy-preserving record linkage with PP-Join
  - Cryptographic bloom filters
  - Privacy-Preserving PP-Join (P4Join)
  - GPU-based implementation
- Big Graph Data
  - Graph-based Business Intelligence with BIIIG
  - GraDoop: Hadoop-based data management and analysis
- Summary and outlook
Facebook
ca. 1.3 Billion users
c. 340 friends per user
Twitter
ca. 300 Million users
c. 500 Million Tweets per day

Internet
ca. 2.9 Billion Users

Gene (human)
20,000-25,000
c. 4 Million individuals
Patients
> 18 Millionen (Germany)
Illnesses
> 30,000

World Wide Web
ca. 1 Billion Websites
LOD-Cloud
c. 31 Billion Triples
- Business intelligence usually based on relational data warehouses
  - enterprise data is integrated within dimensional schema
  - analysis limited to predefined relationships
  - no support for relationship-oriented data mining

- Graph-based approach (BIIIG)
  - Integrate data sources within an instance graph by preserving original relationships between data objects (transactional and master data)
  - Determine subgraphs (business transaction graphs) related to business activities
  - Analyze subgraphs or entire graphs with aggregation queries, mining relationship patterns, etc.
BIIIG DATA INTEGRATION AND ANALYSIS WORKFLOW

„Business Intelligence on Integrated Instance Graphs“

1. Metadata Acquisition
2. Automated Graph Integration
SCREENSHOT FOR NEO4J IMPLEMENTATION
Relational database systems
- store vertices and edges in tables
- utilize indexes, column stores, etc.

Graph database system, e.g. Neo4J
- use of property graph data model: vertices and edges have arbitrary set of properties (represented as key-value pairs)
- focus on simple transactions and queries

Distributed graph processing systems, e.g., Google Pregel, Apache Giraph, GraphX, etc.
- In-memory storage of graphs in Shared Nothing cluster
- parallel processing of general graph algorithms, e.g. page rank, connected components, ...
A comprehensive framework and research platform for efficient, distributed and domain independent graph analytics.

WHAT’S MISSING?
- Hadoop-based framework for graph data management and analysis
- Graph storage in scalable distributed store, e.g., HBase
- Extended property graph data model
  - operators on graphs and sets of (sub) graphs
  - support for semantic graph queries and mining
- Leverages powerful components of Hadoop ecosystem
  - MapReduce, Giraph, Spark, Pig, Drill ...
- New functionality for graph-based processing workflows and graph mining
GRADOOP – HIGH LEVEL ARCHITECTURE

Graph Visualization  Visual Workflow Definition

Graph Operators

Pipeline Execution

Backend

Control

Data

Bulk Load

Graph Repository

Bulk Write

Distributed Storage

Shared Nothing Cluster

Graph Visualization

Visual Workflow Definition

Graph Operators

Pipeline Execution

Backend

Control

Data

Bulk Load

Graph Repository

Bulk Write

Distributed Storage

Shared Nothing Cluster
Partitioned Directed Labeled Attributed Multigraph
# GRADOOP OPERATORS

<table>
<thead>
<tr>
<th>Operator</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>$\gamma: \mathcal{G} \rightarrow (\mathbb{R} \cup \Sigma)$</td>
<td>$G \mapsto g$</td>
</tr>
<tr>
<td>Subgraph Discovery</td>
<td>$\theta_{\nu, \epsilon}: \mathcal{G} \rightarrow \mathcal{G}$</td>
<td>$G \mapsto \mathcal{G}$</td>
</tr>
<tr>
<td>Selection</td>
<td>$\sigma_{\varphi}: \mathcal{G} \rightarrow \mathcal{G}$</td>
<td>Graph set $\mathcal{G}$</td>
</tr>
<tr>
<td>Similarity</td>
<td>$\sim: \langle \mathcal{G}_1, \mathcal{G}_2 \rangle \rightarrow \mathbb{R}$</td>
<td>Graphs $\mathcal{G}_1, \mathcal{G}_2$</td>
</tr>
<tr>
<td>Frequent Subgraphs</td>
<td>$\phi_t: \mathcal{G} \rightarrow \mathcal{G}$</td>
<td>Graph set $\mathcal{G}$</td>
</tr>
</tbody>
</table>

- Summarization
- Pattern Match
- Projection
- Map
- Union
- Intersect
- Difference
- Edit Steps
- Equivalence
- Equality
- Inner Join
- Outer Join
IMPLEMENTATION STATUS

Selection Aggregation

- Gradoop-core
- Gradoop-MapReduce
- Gradoop-Giraph
- Giraph 1.1.0

I/O Formats

- Hadoop 1.2.1
- Hbase 0.98.7

BTG Analysis Pipeline
Data Import

Subgraph Discovery
I/O Formats

EPG Model
HBaseGraphStore
Bulk Load
I/O Formats
BIIIG WITH GRADOOP

Foodbroker
Integrated
Instance Graph
- ScaDS Dresden/Leipzig
- Big Data Integration
  - Introduction
  - Matching product offers from web shops
  - DeDoop: Deduplication with Hadoop
- Privacy-preserving record linkage with PP-Join
  - Cryptographic bloom filters
  - Privacy-Preserving PP-Join (P4Join)
  - GPU-based implementation
- Big Graph Data
  - Graph-based Business Intelligence with BIIIG
  - GraDoop: Hadoop-based data management and analysis
- Summary and outlook
ScaDS Dresden/Leipzig

- Research focus on data integration, knowledge extraction, visual analytics
- Broad application areas (scientific + business-related)
- Solution classes for applications with similar requirements

Big Data Integration

- Big data poses new requirements for data integration (variety, volume, velocity, veracity)
- Comprehensive data preprocessing and cleaning
- Hadoop-based approaches for improved scalability, e.g. Dedoop
- Usability: machine-learning approaches, GUI, ...
Scalable Privacy-Preserving Record Linkage
- bloom filters allow simple, effective and relatively efficient match approach
- Privacy-preserving PP-Join (P4JOIN) achieves comparable performance to multibit trees but easier to parallelize
- GPU version achieves significant speedup
- further improvements needed to reduce quadratic complexity

Big Graph Data
- high potential of graph analytics even for business data (BIIIG)
- GraDoop: infrastructure for entire processing pipeline: graph acquisition, storage, integration, transformation, analysis (queries + graph mining), visualization
- leverages Hadoop ecosystem including graph processing systems
- extended property graph model with powerful operators
Parallel execution of more diverse data integration workflows for text data, image data, sensor data, etc.
- learning-based configuration to minimize manual effort (active learning, crowd-sourcing)

Holistic integration of many data sources (data + metadata)
- clustering across many sources
- N-way merging of related ontologies (e.g. product taxonomies)

Improved privacy-preserving record linkage
- better scalability, also for n-way (multi-party) PPRL

Big Graph data management
- complete processing framework
- improved usability
REFERENCES

- A. Petermann, M. Junghanns, R. Müller, E. Rahm: *BIIIG: Enabling Business Intelligence with Integrated Instance Graphs*. Proc. 5th Int. Workshop on Graph Data Management (GDM 2014)