ScaDS
DRESDEN LEIPZIG

BIG DATA INTEGRATION AT SCADS DRESDEN/LEIPZIG

ERHARD RAHM, UNIV. LEIPZIG

ScaDS DRESDEN LEIPZIG

GERMAN CENTERS FOR BIG DATA

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

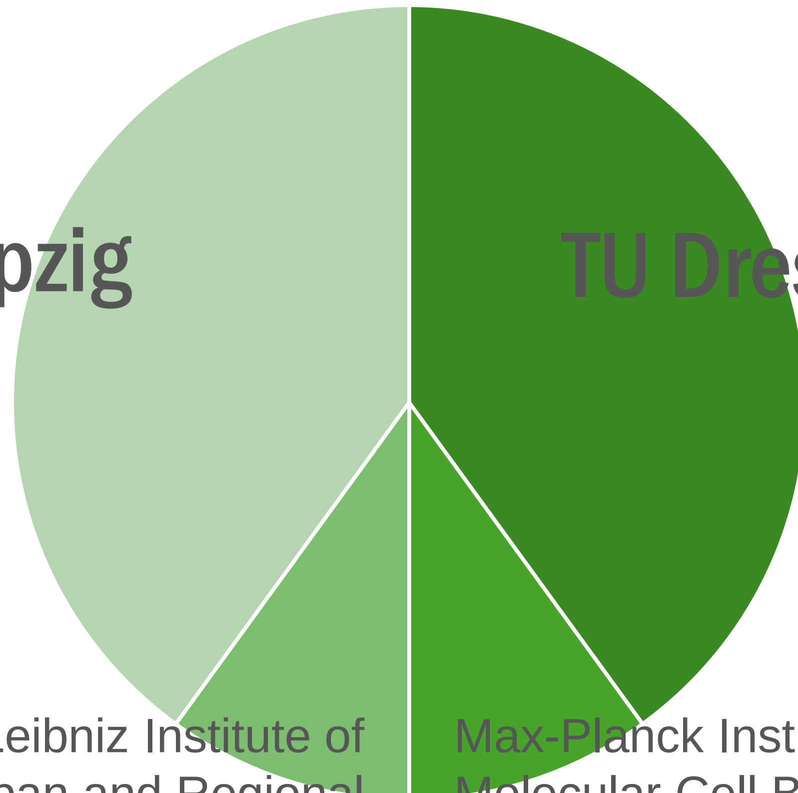
ScaDS DRESDEN LEIPZIG

GOALS

- Bundling and advancement of existing expertise on Big Data
- Development of Big Data Services and Solutions
- Big Data Innovations

ScaDS DRESDEN LEIPZIG

FUNDED INSTITUTES



ScaDS DRESDEN LEIPZIG

ASSOCIATED PARTNERS

- 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

ScaDS DRESDEN LEIPZIG

GROBSTRUKTUR DES ZENTRUMS

- Life sciences
- Material and Engineering sciences
- Environmental / Geo sciences
- Digital Humanities
- Business Data

Service center

Big Data Life Cycle Management and Workflows

Data Quality / Data Integration

Knowledge Extraktion

Visual Analytics

Efficient Big Data Architectures

ScaDS DRESDEN LEIPZIG

RESEARCH PARTNERS

- 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










ScaDS DRESDEN LEIPZIG

APPLICATION CORRDINATORS

- Life sciences G. Myers
- Material / Engineering sciences M. Gude
- Environmental / Geo sciences J. Schanze


- Digital Humanities G. Heyer
- Business Data B. Franczyk

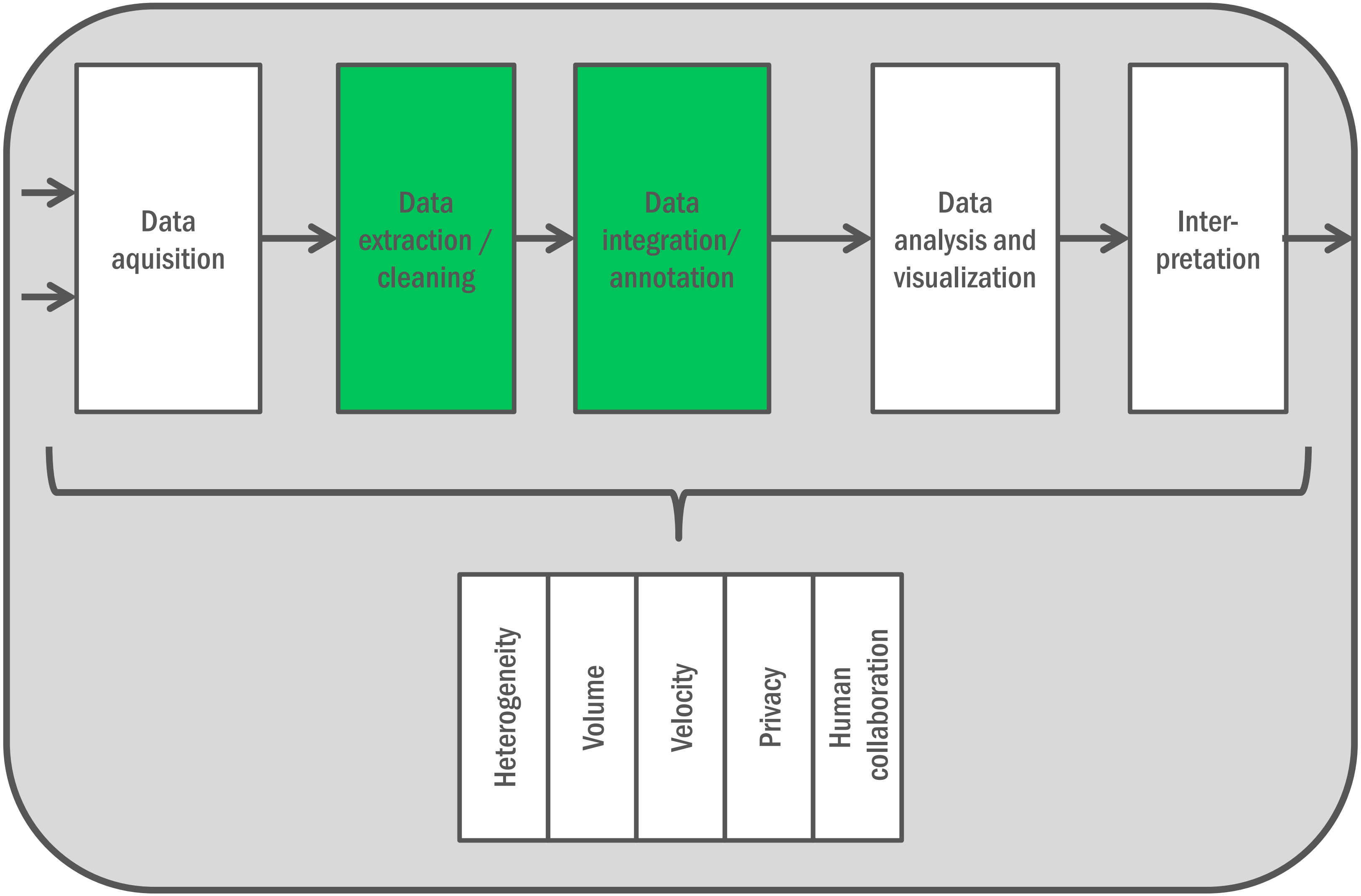

ScaDS DRESDEN LEIPZIG

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
- Summary and outlook
- References

ScaDS DRESDEN LEIPZIG

BIG DATA ANALYSIS PIPELINE



ScaDS DRESDEN LEIPZIG

OBJECT MATCHING (DEDUPLICATION)

- Identification of semantically equivalent objects
  - within one data source or between different sources
- Original focus on structured (relational) data, e.g. customer data

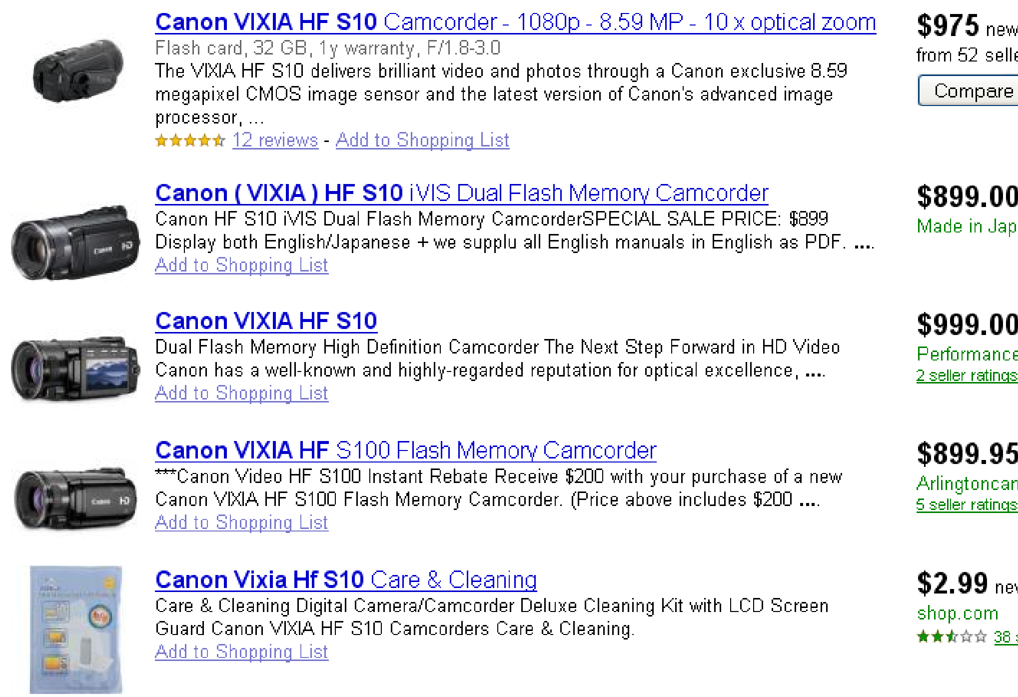
Cno	*LastName*	*FirstName*	*Gender*	*Address*	*Phone/Fax*
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

CID	*Name*	*Street*	*City*	*Sex*
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

ScaDS DRESDEN LEIPZIG

BIG DATA INTEGRATION USE CASE
INTEGRATION OF PRODUCT OFFERS IN COMPARISON PORTAL

- Thousands of data sources (shops/merchants)
- Millions of products and product offers
- Continous changes
- Many similar, but different products
- Low data quality



ScaDS DRESDEN LEIPZIG

USE OF PRODUCT CODES

- Frequent existence of specific product codes for certain products
- Product code = manufacturer-specific identifier
  - any sequence consisting of alphabetic, special, and numeric characters split by an arbitrary number of white spaces.

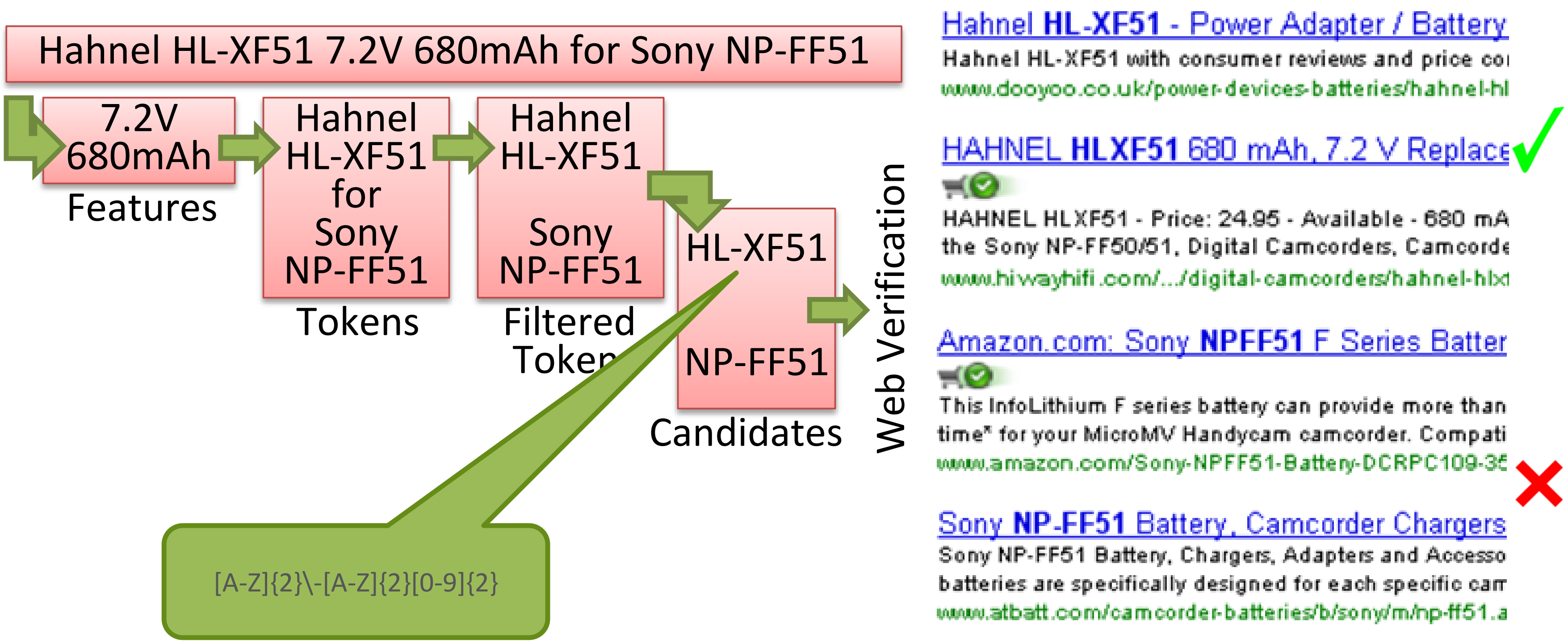
➢ Utilize to differentiate similar but different products.

Canon VIXIA HF S100 Camcorder - 1080p - 8.59 MP

Hahnel HL-XF51 7.2V 680mAh for Sony NP-FF51

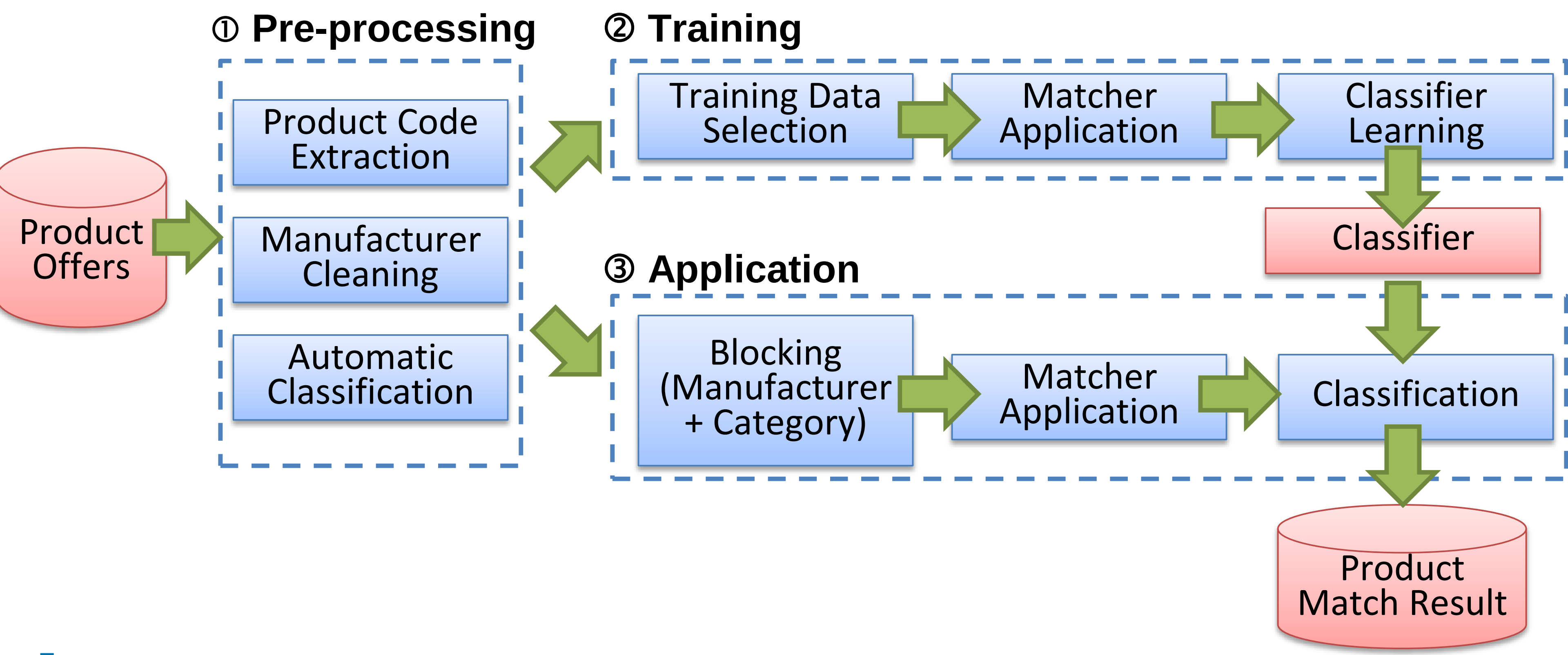
ScaDS DRESDEN LEIPZIG

PRODUCT CODE EXTRACTION



ScaDS DRESDEN LEIPZIG

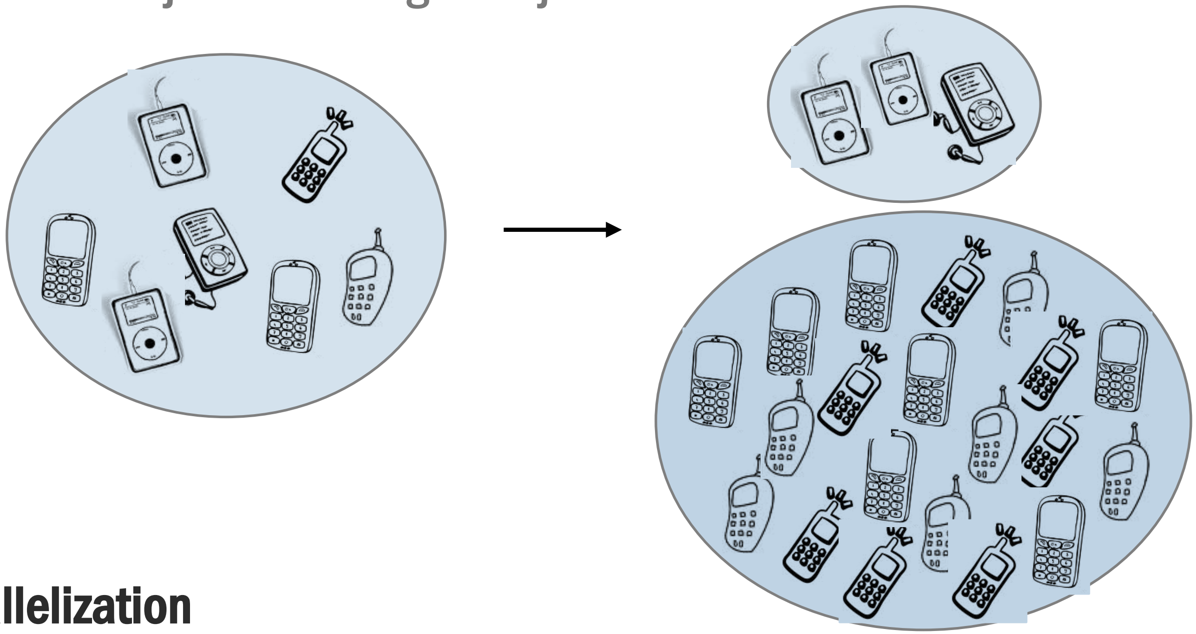
LEARNING-BASED MATCH APPROACH



ScaDS DRESDEN LEIPZIG

HOW TO SPEED UP OBJECT MATCHING?

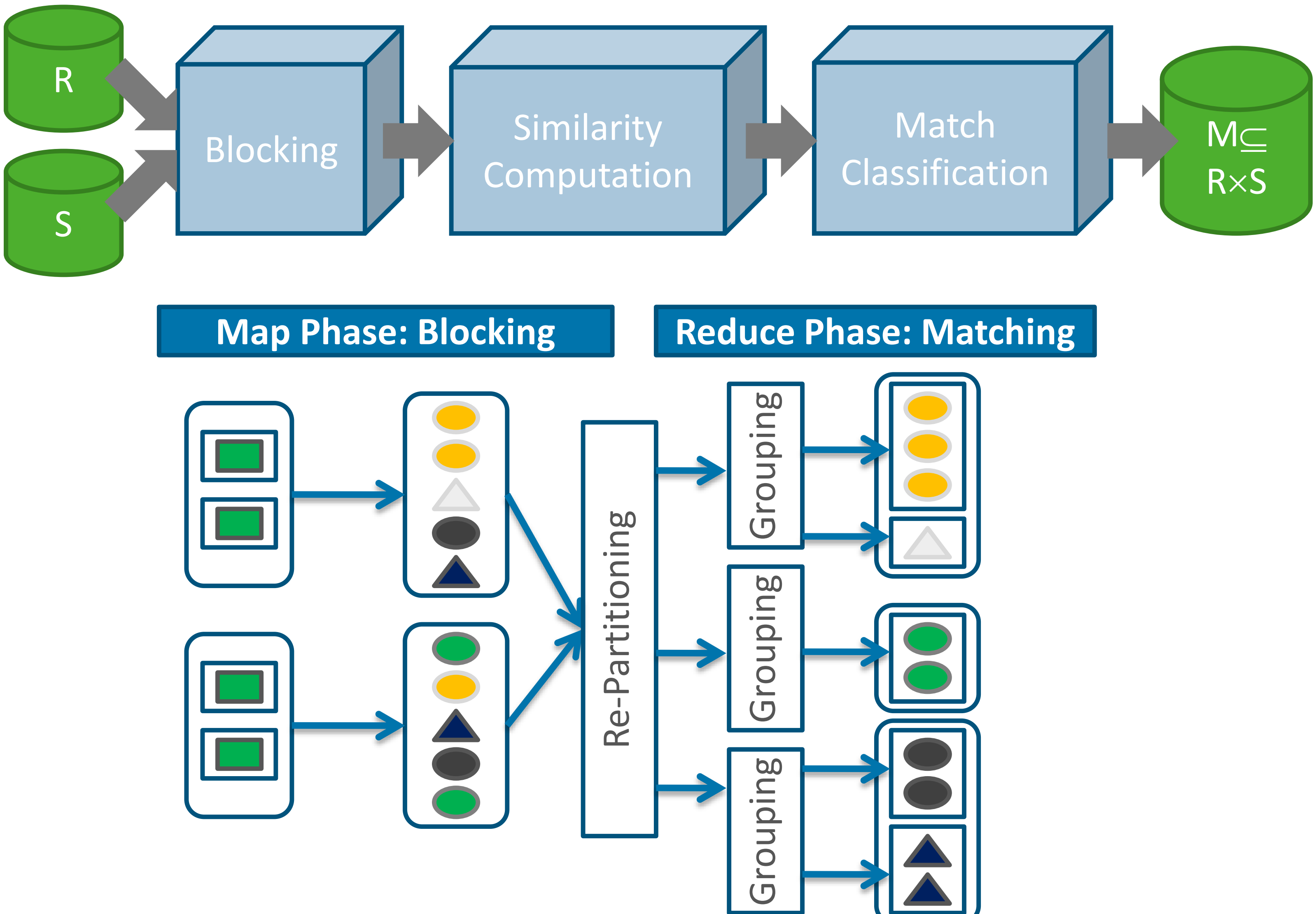
- **Blocking** to reduce search space
  - group similar objects within blocks based on *blocking key*
  - restrict object matching to objects from the same block



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

ScaDS DRESDEN LEIPZIG

GENERAL OBJECT MATCHING WORKFLOW



ScaDS DRESDEN LEIPZIG

LOAD BALANCING

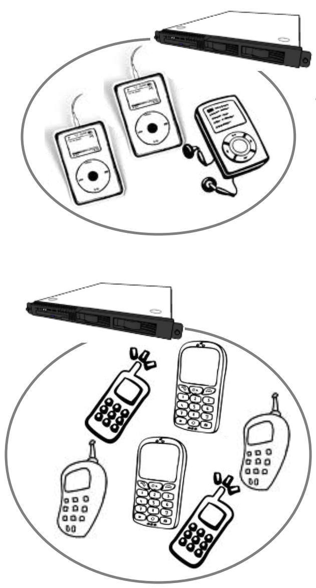
- Data skew leads to unbalanced workload
  - Large blocks prevent utilization of more than a few nodes
  - Deteriorates scalability and efficiency
  - Unnecessary costs (you also pay for underutilized machines!)

- Key ideas for load balancing
  - Additional MR job to determine blocking key distribution, i.e., number and size of blocks (per input partition)
  - Global load balancing that assigns (nearly) the same number of pairs to reduce tasks
- Simplest approach : BlockSplit (ICDE2012)
  - split large blocks into sub-blocks with multiple match tasks
  - distribute the match tasks among multiple reduce tasks

BLOCK SPLIT: 1 SLIDE ILLUSTRATION

- Example: 3 MP3 players + 6 cell phones → 18 pairs (1 time unit)
- Parallel matching on 2 (reduce) nodes

naiive approach

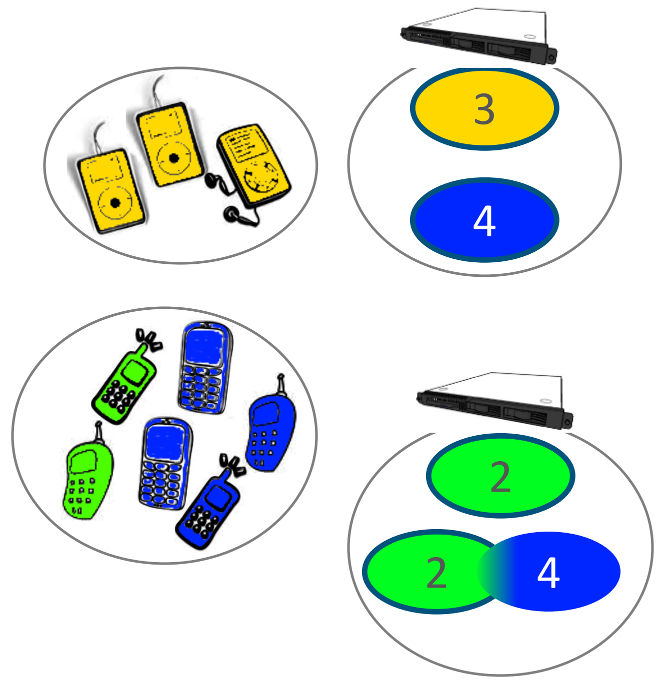


3 pairs
(16%)

15 pairs
(84%)

Speedup: 18/15=1.2

BlockSplit



3 pairs
<u>6 pairs</u>
9 pairs (50%)

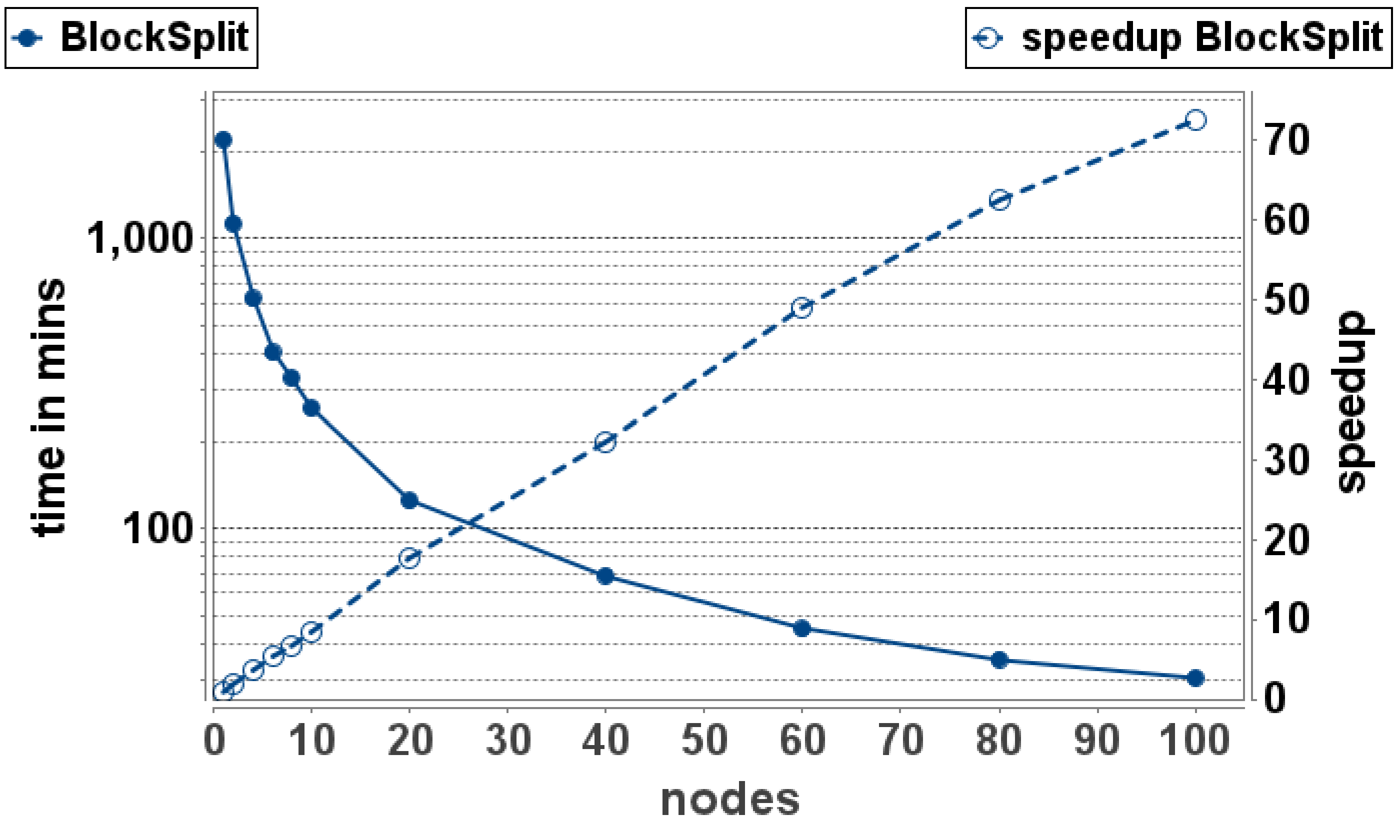
1 pair
<u>8 pairs</u>
9 pairs (50%)

Speedup: 2

ScaDS DRESDEN LEIPZIG

BLOCK SPLIT EVALUATION: SCALABILITY

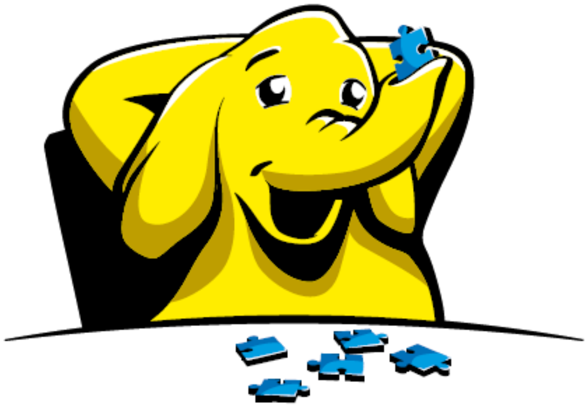
- Evaluation on Amazon EC infrastructure using Hadoop
- Matching of 114.000 product records



ScaDS DRESDEN LEIPZIG

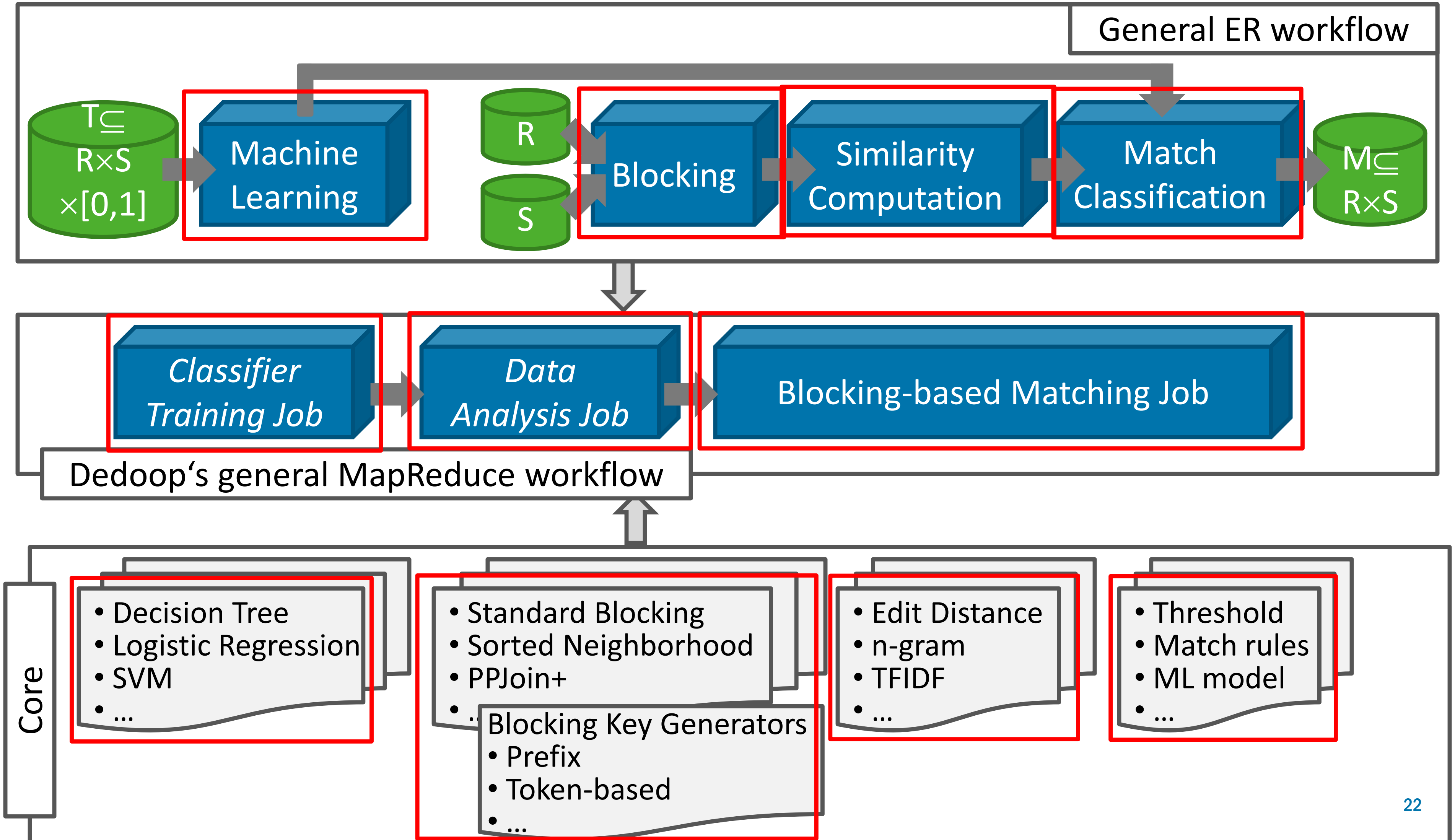
DEDOOP: EFFICIENT DEDUPLICATION WITH HADOOP

- 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



ScaDS DRESDEN LEIPZIG

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
- Summary and outlook
- References

PRIVACY-PRESERVING RECORD LINKAGE

- Object matching with encrypted data to preserve privacy
  - data exchange / integration for person-related data
  - many use cases: medicine (e.g., cancer registries), census, …
- numerous PPRL approaches (Vatsalan et al., 2013), some requiring trustee or secure multi-party protocol
- scalability problem for large datasets (e.g., for census purposes)

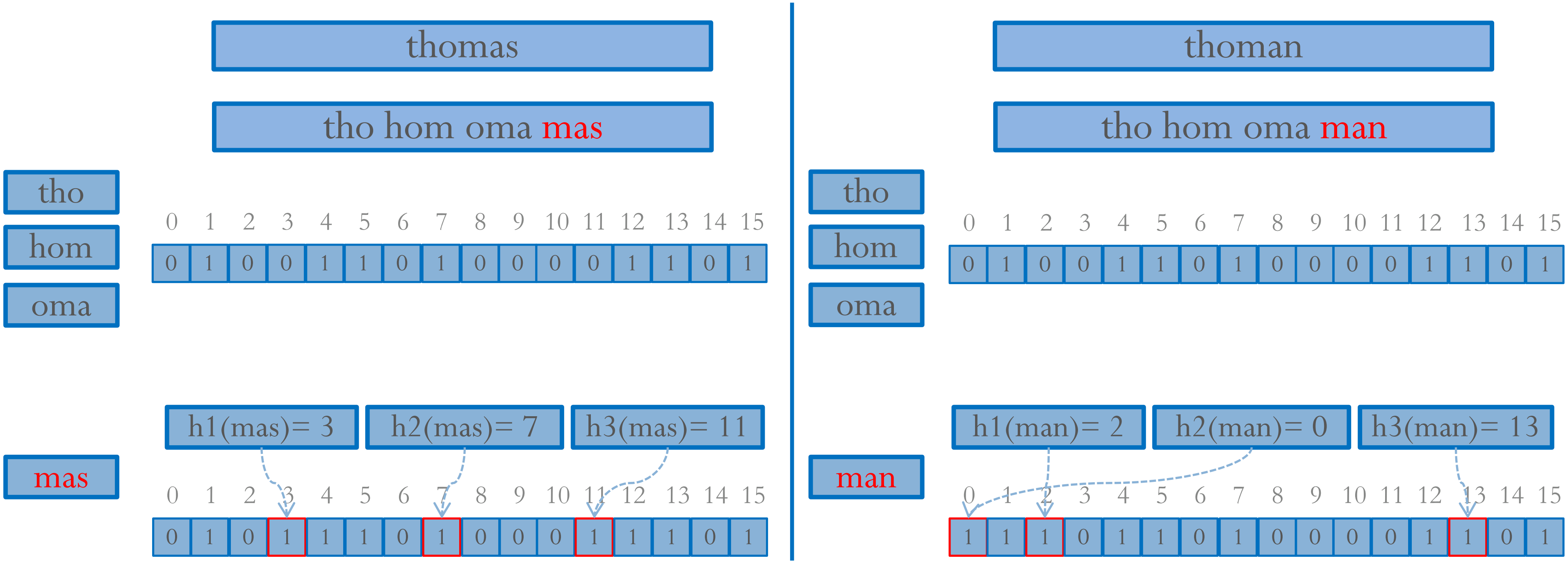
ScaDS DRESDEN LEIPZIG

PPRL WITH BLOOM FILTERS

- effective and simple approach 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 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

ScaDS DRESDEN LEIPZIG

SIMILARITY COMPUTATION - EXAMPLE



$$Sim_{Jaccard}(r1, r2) = (r1 \wedge r2) \;/\; (r1 \vee r2)$$

$$Sim_{Jaccard}(r1, r2) = 7/11$$

ScaDS DRESDEN LEIPZIG

PP-JOIN: POSITION PREFIX JOIN (XIAO ET AL, 2008)

- one of the most efficient *similarity join* algorithms
  - determine all pairs of records with $sim_{Jaccard}(x,y) \geq t$
- 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

ScaDS DRESDEN LEIPZIG

LENGTH FILTER

- matching records pairs must have similar lengths

$$Sim_{Jaccard}(x, y) \geq t \Rightarrow |x| \geq |y| * t$$

- length / cardinality: number of tokens for strings, number of bits for bit vectors

- Example for minimal similarity *t* = 0,8:

records	tokens (bigrams)	length
r1=Tom	[_t, to, om, m_]	4
r2=Thomas	[_t, th, ho, om, ma, as, s_]	7
r4=Tommy	[_t, to, om, mm, my, y_]	6

$\vert \mathbf{r1} \vert \geq \vert \mathbf{r2} \vert * t$?	$4 \geq 5{,}6$?	no
$\vert \mathbf{r4} \vert \geq \vert \mathbf{r2} \vert * t$?	$6 \geq 5{,}6$?	yes

ScaDS DRESDEN LEIPZIG

PREFIX FILTER

- Similar records must have a minimal overlap α in their sets of tokens (or set bit positions)

$$\mathrm{Sim}_{\mathrm{Jaccard}}(\mathrm{x},\mathrm{y}) \geq t \quad \Leftrightarrow \quad \mathit{Overlap}(\mathrm{x},\mathrm{y}) \geq \alpha = \left\lceil \left(\frac{t}{1+t} * (|x|) + |y|) \right) \right\rceil$$

- Prefix filter approximates this test
  - order all tokens (bit positions) for all records according to their overall frequency from infrequent to frequent
  - exclude pairs of records without any overlap in their prefixes with

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

- Example (t= 0.8)

records	sorted tokens	prefix length	Prefix
r2=Thomas	[ho, th, ma, as, s_, _t, om]	3	[ho, th, ma]
r3=Tomas	[ma, as, s_, to, _t, om]	3	[ma, as, s_]
r4=Tommy	[mm, my, y_, to, -t, om]	3	[mm, my, y_]

$\mathrm{prefix(r2)} \cap \mathrm{prefix(r3)} = \{\mathrm{ma}\} \neq \{\}$

$\mathrm{prefix(r2)} \cap \mathrm{prefix(r4)} = \{\}$

$\mathrm{prefix(r3)} \cap \mathrm{prefix(r4)} = \{\}$

ScaDS DRESDEN LEIPZIG

PRIVACY-PRESERVING PP-JOIN (P4JOIN)

- evaluate overlap of set positions in bit vectors
- Preprocessing phase
  - determine frequency per bit positions and reorder all bit vectors according to the overall frequency of bit positions
  - determine length and prefix per bit vector
  - sort all bit vectors in ascending order of their „length“ (number of set positions)
- Match phase (sequential scan)
  - for each record apply *length filter* to determine window of relevant records to match with
  - apply *prefix filter* (AND operation on prefix) to exclude record pairs without prefix overlap
  - apply *position filter* for further savings

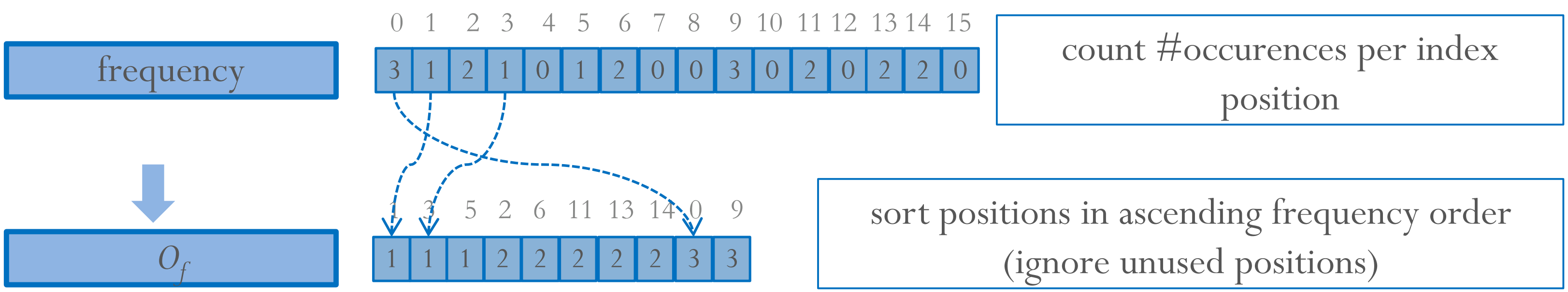
ScaDS DRESDEN LEIPZIG

P4JOIN: PREPROCESSING (1)

- **records** (id, bit vector/ fingerprint)

ID	fingerprint	card.	tokens (set positions)
A	1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0	8	0 2 3 6 9 11 13 14
B	1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0	4	0 1 5 9
C	1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0	7	0 2 6 9 11 13 14

- determine *frequency ordering* O_f



ScaDS DRESDEN LEIPZIG

PPPP-JOIN: PREPROCESSING (2)

- reorder fingerprints according to O_f

ID	fingerprint	card.	tokens (set position)	sorted tokens
A	1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0	8	0 2 3 6 9 11 13 14	3 2 6 11 13 14 0 9
B	1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0	4	0 1 5 9	1 5 0 9
C	1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0	7	0 2 6 9 11 13 14	2 6 11 13 14 0 9

O_f	1	3	5	2	6	11	13	14	0	9
	1	1	1	2	2	2	2	2	3	3

continue with reordered fingerprints

ID	reordered fingerprint	card.
A	0 1 0 1 1 1 1 1 1 1 0 0 0	8
B	1 0 1 0 0 0 0 0 1 1 0 0 0	4
C	0 0 0 1 1 1 1 1 1 1 0 0 0	7
	1 3 5 2 6 11 13 14 0 9	

ScaDS DRESDEN LEIPZIG

P4JOIN: PREPROCESSING (3)

- sort records by length (cardinality) and determine prefixes

$$\textbf{prefix\_length(x)} = \lceil ((1-t)*|\mathbf{x}|) + 1 \rceil$$

ID	reordered fingerprint	card.	prefix length	prefix fingerprint
B	1 0 1 0 0 0 0 0 1 1 0 0 0 0	4	2	1 0 1
C	0 0 0 1 1 1 1 1 1 1 0 0 0 0	7	3	0 0 0 1 1 1
A	0 1 0 1 1 1 1 1 1 1 0 0 0 0	8	3	0 1 0 1 1

ScaDS DRESDEN LEIPZIG

P4JOIN: APPLY LENGTH FILTER

- compare records ordered by length

ID	reordered fingerprint	card.
B	1 0 1 0 0 0 0 0 1 1 0 0 0	4
C	0 0 0 1 1 1 1 1 1 1 0 0 0	7
A	0 1 0 1 1 1 1 1 1 1 0 0 0	8

length filter
$7 * 0.8 = 5.6$

when reading record C it is observed that it does not meet the length filter w.r.t. B -> record B (|B|=4) can be excluded from all further comparisons

ID	reordered fingerprint	card.
B	1 0 1 0 0 0 0 0 1 1 0 0 0	4
C	0 0 0 1 1 1 1 1 1 1 0 0 0	7
A	0 1 0 1 1 1 1 1 1 1 0 0 0	8

length filter

$8 * 0.8 = 6.4$

record A still needs to be considered w.r.t. C due to similar length

ScaDS DRESDEN LEIPZIG

P4JOIN: PREFIX FILTER

- only records with overlapping prefix need to be matched
  - AND operation on prefix fingerprints

ID	reordered fingerprint	card.	prefix fingerprint
B	1 0 1 0 0 0 0 0 1 1 0 0 0 0	4	1 0 1
C	0 0 0 1 1 1 1 1 1 1 0 0 0 0	7	0 0 0 1 1 1
A	0 1 0 1 1 1 1 1 1 1 0 0 0 0	8	0 1 0 1 1

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

ScaDS DRESDEN LEIPZIG

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 α* 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
    < minimal overlap α = 8

ScaDS DRESDEN LEIPZIG

EVALUATION

- 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, ..., 500.000]. $|R|=1/5 \cdot N$, $|S|=4/5 \cdot N$
  - bit vector length: 1000
  - similarity threshold 0.8

ScaDS DRESDEN LEIPZIG

EVALUATION RESULTS

- runtime in minutes on standard PC

Approach	Dataset size N				
	100.000	200.000	300.000	400.000	500.000
NestedLoop	6,10	27,68	66,07	122,02	194,77
MultiBitTree	4,68	18,95	40,63	78,23	119,73
P4 Length filter only	3,38	20,53	46,48	88,33	140,73
P4 Length+Prefix	3,77	22,98	52,95	99,72	159,22
P4 Length+Prefix+Position	**2,25**	**15,50**	**40,05**	**77,80**	125,52

- similar results for P4Join and Multibit Tree
- relatively small improvements compared to NestedLoop

ScaDS DRESDEN LEIPZIG

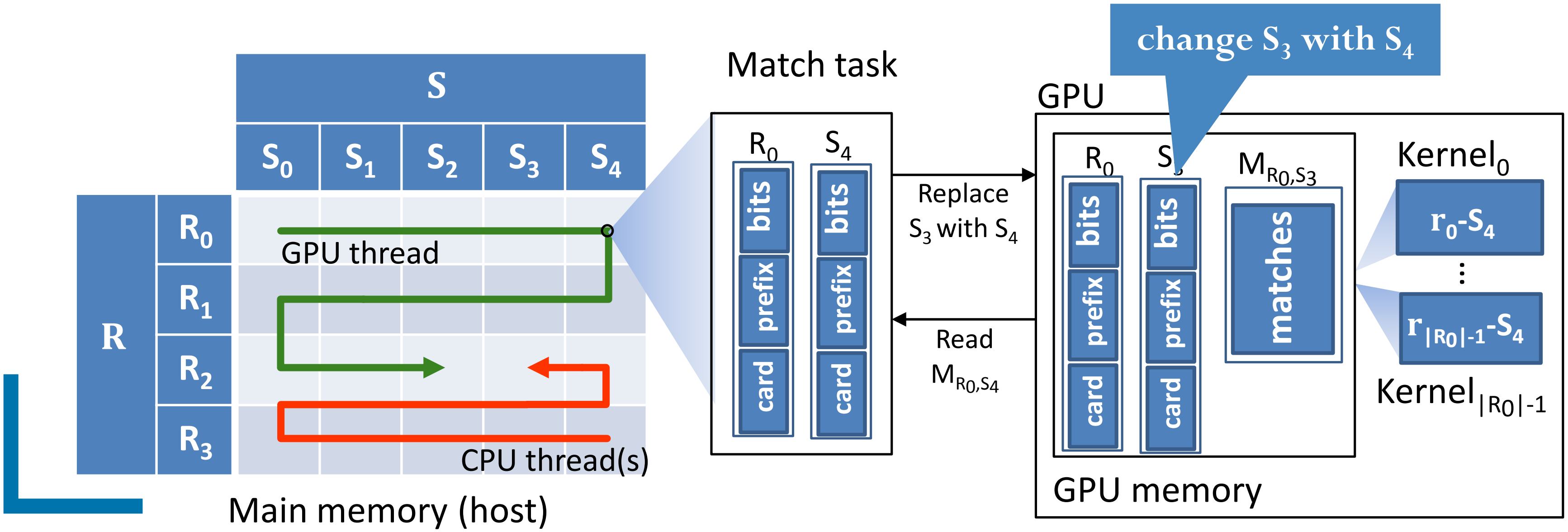
GPU-BASED PPRL

- 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

ScaDS DRESDEN LEIPZIG

EXECUTION SCHEME

- 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

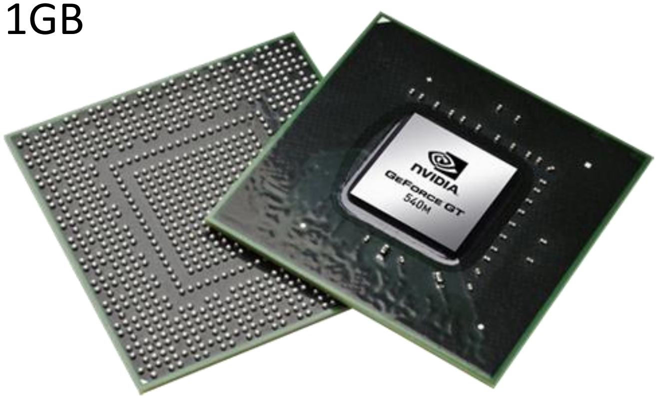


ScaDS DRESDEN LEIPZIG

GPU-BASED EVALUATION RESULTS

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

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



	100.000	200.000	300.000	400.000	500.000
GForce GT 610	0,33	1,32	2,95	5,23	8,15
GeForce GT 540M	0,28	1,08	2,41	4,28	6,67

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

ScaDS DRESDEN LEIPZIG

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
- Summary and outlook
- References

ScaDS DRESDEN LEIPZIG

SUMMARY

- 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 preprocessinga and cleaning
  - Hadoop-based approaches for improved scalability, e.g. Dedoop
  - Usabilty: machine-learning approaches, GUI, monitoring ...

ScaDS DRESDEN LEIPZIG

SUMMARY (2)

- Privacy-Preserving Record Linkage
  - increasingly important tp protect personal information
  - Scalability issues for Big Data
  - Bloom filters allow simple, effective and relatively efficient match approach
  - still scalability issues for Big Data -> reduce search space and apply parallel processing
- Privacy-preserving PP-Join (P4JOIN)
  - relatively easy adoption for bit vectors with improved position filter
  - comparable performance to Multibit trees but easier to parallelize
  - GPU version achieves significant speedup
  - further improvements needed to reduce quadratic complexity

ScaDS DRESDEN LEIPZIG

OUTLOOK

- 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)
- Realtime data enrichment and integration for sensor data
- Improved privacy-preserving record linkage

ScaDS DRESDEN LEIPZIG

REFERENCES

- H. Köpcke, A. Thor, S. Thomas, E. Rahm: *Tailoring entity resolution for matching product offers*. Proc. EDBT 2012: 545-550
- L. Kolb, E. Rahm: *Parallel Entity Resolution with Dedoop*. Datenbank-Spektrum 13(1): 23-32 (2013)
- L. Kolb, A. Thor, E. Rahm: Dedoop: *Efficient Deduplication with Hadoop*. PVLDB 5(12), 2012
- L. Kolb, A. Thor, E. Rahm: *Load Balancing for MapReduce-based Entity Resolution*. ICDE 2012: 618-629
- L. Kolb, Z. Sehili, E. Rahm: *Iterative Computation of Connected Graph Components with MapReduce*. Datenbank-Spektrum 14(2): 107-117 (2014)
- E. Rahm, W.E. Nagel: *ScaDS Dresden/Leipzig: Ein serviceorientiertes Kompetenzzentrum für Big Data*. Proc. GI-Jahrestagung 2014: 717
- R.Schnell, T. Bachteler, J. Reiher: *Privacy-preserving record linkage using Bloom filters*. BMC Med. Inf. & Decision Making 9: 41 (2009)
- Z. Sehili, L. Kolb, C. Borgs, R. Schnell, E. Rahm: *Privacy Preserving Record Linkage with PPJoin*. Proc. BTW Conf. 2015 (to appear)
- D. Vatsalan, P. Christen, V. S. Verykios: *A taxonomy of privacy-preserving record linkage techniques*. Information Syst. 38(6): 946-969 (2013)
- C. Xiao, W. Wang, X. Lin, J.X. Yu: *Efficient Similarity Joins for Near Duplicate Detection*. Proc. WWW 2008