

Large-Scale Data Analytics

Aris Gkoulalas-Divanis • Abderrahim Labbi
Editors

Large-Scale Data Analytics

Editors

Aris Gkoulalas-Divanis
IBM Research – Ireland
Damastown Industrial Estate
Mulhuddart, Ireland

Abderrahim Labbi
IBM Research – Zurich
Rüschlikon, Switzerland

ISBN 978-1-4614-9241-2 ISBN 978-1-4614-9242-9 (eBook)

DOI 10.1007/978-1-4614-9242-9

Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2013954541

© Springer Science+Business Media New York 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

In recent years, we are witnessing a data explosion: almost 90 % of today's data have been produced only in the last 2 years, with data being nowadays produced in the order of Zettabytes! This data comes from various sources, including sensors, social networking sites, mobile phone applications, electronic medical record systems and e-commerce sites, just to name a few. Apart from its massive volume, this data is also characterized by variety (heterogeneity) and velocity (streams of data).

Traditional approaches and algorithms are not able to process and analyze such massive and complex datasets. This has signified the need for a paradigm shift, where new hardware and software technology is emerging to efficiently and reliably manage, store, process, analyze and synthesize very large amounts of complex data generated by massively distributed data sources. Beside their massively distributed nature, which requires new distributed architectures for data analysis, the heterogeneity of such sources imposes significant challenges for the efficient analysis of the data under numerous constraints, such as consistent data integration, data homogenization and scaling, privacy and security preservation. Moreover, the emerging real-world applications in domains such as healthcare, weather forecasting, financial engineering, urban planning, traffic management and environmental monitoring impose extra requirements for large-scale data analysis.

This edited book contains contributions on cutting edge research related to large-scale data analytics in the following core areas: databases, data mining, supercomputing, data visualization and privacy. Our goal is to present to students, researchers, professionals and practitioners the state-of-the-art research, which will help shape up the future of large-scale analytics, leading the way to the design of new approaches and technologies that can analyze and synthesize very large amounts of heterogeneous data, generated by massively distributed data sources.

Each chapter of the book presents a survey of an area in large-scale data analytics, or individual results of the emerging research in the field. Chapters 1 and 2 are devoted to the MapReduce framework. In particular, the first chapter provides a comprehensive survey for a family of approaches and mechanisms of large scale data analysis that have been implemented based on the MapReduce framework. Chapter 2 focuses on optimization approaches for plain MapReduce

jobs, as well as for parallel data flow systems. Chapters 3 and 4 present two important application areas of the MapReduce framework: mining tera-scale graphs for patterns and anomalies (Chap. 3), and analyzing customer behavioral data for the Telecom industry (Chap. 4). In Chap. 5, the authors describe a unified heterogeneous architecture that integrates massively threaded shared-memory multiprocessors into MapReduce-based clusters to enable executing Map and Reduce operators on thousands of threads, across multiple GPU devices and nodes. The proposed hybrid system can be used to accelerate machine learning algorithms, such as support vector machines, achieving significant speedup. Chapter 6 is devoted to large-scale social network analysis, offering a comprehensive survey of the state-of-the-art in this area, with focus on parallel algorithms and libraries for the computation of network centrality metrics. An overview of data visualization methods that help users to gain insight into large, heterogeneous, dynamic textual datasets is provided in Chap. 7. The last chapter of the book is devoted to technologies for offering security and privacy at large scale. The authors of this chapter present a novel framework for privacy-preserving, distributed data analysis that is practical for many real-world applications.

We, as editors, are genuinely grateful to all contributors of this book for the time and effort they put into this project, despite the heavy burden that we put on them. We also owe special thanks to the effort of the external reviewers for their help in this effort. Last but not least, we are indebted to Susan Lagerstrom-Fife and Courtney Clark from Springer, for their great support towards the preparation and completion of this work. Their editing suggestions were valuable to improving the organization, readability and appearance of the manuscript.

Mulhuddart, Ireland
Rüschlikon, Switzerland

Aris Gkoulalas-Divanis
Abderrahim Labbi

Contents

1	The Family of Map-Reduce	1
	Sherif Sakr and Anna Liu	
2	Optimization of Massively Parallel Data Flows	41
	Fabian Hueske and Volker Markl	
3	Mining Tera-Scale Graphs with “Pegasus”: Algorithms and Discoveries	75
	U Kang and Christos Faloutsos	
4	Customer Analyst for the Telecom Industry	101
	David Konopnicki and Michal Shmueli-Scheuer	
5	Machine Learning Algorithm Acceleration Using Hybrid (CPU-MPP) MapReduce Clusters	129
	Sergio Herrero-Lopez and John R. Williams	
6	Large-Scale Social Network Analysis	155
	Mattia Lambertini, Matteo Magnani, Moreno Marzolla, Danilo Montesi, and Carmine Paolino	
7	Visual Analysis and Knowledge Discovery for Text	189
	Christin Seifert, Vedran Sabol, Wolfgang Kienreich, Elisabeth Lex, and Michael Granitzer	
8	Practical Distributed Privacy-Preserving Data Analysis at Large Scale	219
	Yitao Duan and John Canny	
	Index	253

Contributors

John Canny Computer Science Division, University of California, Berkeley, CA, USA

Yitao Duan NetEase Youdao, Beijing, China

Christos Faloutsos School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Michael Granitzer University of Passau, Passau, Germany

Sergio Herrero-Lopez Technologies, Equities and Currency (TEC) Division, SwissQuant Group AG, Zurich, Switzerland

Fabian Hueske Technische Universität Berlin, Berlin, Germany

U Kang Department of Computer Science, KAIST University, Republic of Korea

Wolfgang Kienreich Know-Center Graz, Graz, Austria

David Konopnicki IBM Haifa Research Lab, Haifa, Israel

Mattia Lambertini Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

Elisabeth Lex Know-Center Graz, Graz, Austria

Anna Liu NICTA and University of New South Wales, Sydney, NSW, Australia

Matteo Magnani Department of Information Technology, Uppsala University, 751 05 Uppsala, Sweden

Volker Markl Technische Universität Berlin, Berlin, Germany

Moreno Marzolla Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

Danilo Montesi Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

Carmine Paolino Department of Computer Science, Vrije Universiteit, Amsterdam, The Netherlands

Vedran Sabol Know-Center Graz, Graz, Austria

Sherif Sakr NICTA and University of New South Wales, Sydney, NSW, Australia

Christin Seifert University of Passau, Passau, Germany

Michal Shmueli-Scheuer IBM Haifa Research Lab, Haifa, Israel

John R. Williams Massachusetts Institute of Technology, Cambridge, MA, USA

List of Figures

Fig. 1.1	Data explosion in scientific computing [26]	2
Fig. 1.2	An example of a MapReduce program [16]	4
Fig. 1.3	An overview of the flow of execution in a MapReduce operation [16]	6
Fig. 1.4	Execution steps of the WordCount example using MapReduce	7
Fig. 1.5	Decision tree for choosing between various join strategies on the MapReduce framework [10]	8
Fig. 1.6	An overview of the Map-Reduce-Merge framework [46]	9
Fig. 1.7	A sample execution of the Map-Reduce-Merge framework [46] ..	10
Fig. 1.8	An overview of the HaLoop architecture [11]	13
Fig. 1.9	Example file co-location in CoHadoop [20]	16
Fig. 1.10	An example SQL query and its equivalent Pig Latin program [23]	18
Fig. 1.11	Pig compilation and execution steps [33]	19
Fig. 1.12	An example of a Sawzall program [35]	20
Fig. 1.13	Basic syntax of SQL/MR query function [22]	21
Fig. 1.14	Two equivalent SCOPE scripts in SQL-like style and in MapReduce-like style [13]	22
Fig. 1.15	The SCOPE/Cosmos execution platform [13]	23
Fig. 1.16	LINQ-expression execution in DryadLINQ [47]	26
Fig. 1.17	A sample Jaql script [9]	27
Fig. 1.18	The Jaql system architecture [9]	28
Fig. 1.19	An example HiveQL query [40]	30
Fig. 1.20	The architecture of HadoopDB [1]	32
Fig. 2.1	The MapReduce programming model	46
Fig. 2.2	The MapReduce execution model	47
Fig. 2.3	Hive partitioning feature: (a) Hive metastore and (b) Hive query execution.....	54
Fig. 2.4	The Starfish architecture	56

Fig. 2.5	A Dryad program DAG and a Dryad communication graph	61
Fig. 2.6	Input contracts: (a) Cross, (b) Match and (c) CoGroup	65
Fig. 2.7	Comparing massively parallel data flow system stacks	68
Fig. 3.1	Radius plot of the YahooWeb graph. Notice the effective diameter is surprisingly small. Also notice the multi-modality, which is possibly due to a mixture of relatively smaller subgraphs	78
Fig. 3.2	Average diameter vs. number of nodes in lin-log scale for the three different Web graphs, where M and B represent millions and billions, respectively. (0.3M): Web pages inside nd.edu at 1999, from Albert et al.'s work [2]. (203M): Web pages crawled by Altavista at 1999, from Broder et al.'s work [6]. (1.4B): Web pages crawled by Yahoo at 2002 (YahooWeb in Table 3.1). The annotations (Albert et al., Sampling, HADI) near the points represent the algorithms for computing the diameter. The Albert et al.'s algorithm seems to be an exact breadth first search, although not clearly specified in their paper. Notice the relatively small diameters for both the directed and the undirected cases. Also notice that the diameters of the undirected Web graphs remain near-constant	79
Fig. 3.3	(a) Static radius plot (count versus radius) of U.S. Patent graph. Notice the bi-modal structure with 'outsiders' (nodes in the disconnected components), 'core' (central nodes in the giant connected component), and 'whiskers' (nodes connected to the giant connected component with long paths). (b) The decomposition of the radius plot using the connected components information. Biggest curve with radius ranging from 11 to 35 the distribution for the giant connected component; small curves on the bottom, left several disconnected components	80
Fig. 3.4	Evolution of the effective diameter of real graphs. The diameter increases until a 'gelling' point, and starts to shrink after the point. (a) Patent. (b) LinkedIn	80
Fig. 3.5	Radius distribution over time. "Expansion": the radius distribution moves to the right until the gelling point. "Contraction": the radius distribution moves to the left after the gelling point. (a) Patent-expansion. (b) Patent-contraction. (c) LinkedIn-expansion. (d) LinkedIn-contraction	81

Fig. 3.6	The evolution of connected components. (a) The giant connected component grows for each year. However, the second largest connected component do not grow above Dunbar's number (≈ 150) and the slope of the size distribution remains constant after the gelling point at year 2003. (b) As in LinkedIn, notice the growth of giant connected component, the size of the second largest connected component bounded above, and the constant slope of the size distribution	82
Fig. 3.7	Connected components size distribution of YahooWeb. Notice the two anomalous spikes which deviate significantly from its neighbors	82
Fig. 3.8	The degree vs. participating triangles of some 'celebrities' in Twitter accounts. Also shown are accounts of adult sites advertisers which have smaller degree, but belong to an abnormally large number of triangles. The reason of the large number of triangles is that adult accounts are often created from the same provider, and they follow each other to form a clique, to possibly boost their rankings or popularity	84
Fig. 3.9	GIM-V NNB. The matrix elements are grouped into 2×2 blocks denoted by $B_{i,j}$. The vector elements are grouped into length 2 blocks denoted by V_i . The matrix and vector are joined block-wise, not element-wise	89
Fig. 3.10	Non-clustered vs. clustered adjacency matrices for two isomorphic graphs. Each node has a self loop which is omitted in the figure for clarity. The edges are grouped into 2 by 2 blocks. The right matrix uses only three blocks while the left matrix uses nine blocks. GIM-V CCB uses the clustered matrix	90
Fig. 3.11	Machine scalability of our proposed CCB method. The Y-axis shows the ratio of the running time T_M with M machines, and T_{25} , for PageRank queries. Note the running time scales up near-linearly with the number of machines	91
Fig. 3.12	Edge scalability of our proposed CCB method. The Y-axis shows the running time in seconds, for PageRank queries on Kronecker graphs. Note the running time scales up near-linearly with the number of edges for all the settings (10, 25, and 40 machines)	91

Fig. 3.13	Effectiveness of our proposed CCB method compared to the naive NNB method. (a) File size comparison after clustering and compression. The Y-axis is in log scale. Note our proposed method reduces the data size up to $43\times$ smaller than the naive method. The ‘Random’ graph has better performance gain than real-world graphs since the density is much higher. (b) Running time comparison of PageRank queries. Our proposed method outperforms the naive method by $9.2\times$ 92	92
Fig. 3.14	Comparison of running time between different skewed matrix-matrix multiplication methods in MapReduce. Our proposed CBMM outperforms naive methods by at least $76\times$. The slowest matrix-matrix multiplication algorithm (MM) even didn’t finish and the job failed due to the excessive amount of data 97	97
Fig. 4.1	The customer analyst library 116	116
Fig. 4.2	(a) Telcom flow using customer analyst, and (b) end-to-end flow 119	119
Fig. 4.3	Percentage of the different match levels 122	122
Fig. 4.4	Runtime performance (in hours) with respect to increasing in number of processed days 123	123
Fig. 5.1	MapReduce primitives and runtime 132	132
Fig. 5.2	Decomposition of K -means into MapReduce tasks 133	133
Fig. 5.3	Decomposition of EM using Gaussian mixtures into MapReduce tasks..... 133	133
Fig. 5.4	Decomposition of SVM into MapReduce tasks 134	134
Fig. 5.5	The MapReduce architecture 138	138
Fig. 5.6	Data node (DN) 139	139
Fig. 5.7	Massively parallel processor node (MPPN)..... 140	140
Fig. 5.8	Port abstraction and its components..... 142	142
Fig. 5.9	Scatter-Gather using ports and MPPs 143	143
Fig. 5.10	Binary SVM decomposed into MapReduce tasks 147	147
Fig. 5.11	Multiple-MPP device SVM..... 148	148
Fig. 6.1	An example of the main centrality measures 161	161
Fig. 6.2	Three graphs with 50 nodes and different structures: from left to right, a random graph (with wiring probability: 0.05), a Watts-Strogatz small world network and a Barabási-Albert free-scale network 162	162
Fig. 6.3	Schematic of shared memory architectures. (a) UMA. (b) NUMA 163	163
Fig. 6.4	Schematic representation of the distributed memory architecture 166	166

Fig. 6.5	Different partitioning strategies may lead to very different performance of the same algorithm. <i>Arrows</i> denote (directed) graph edges	168
Fig. 6.6	Graph replication and partitioning example. (a) Replication. (b) Partitioning	169
Fig. 6.7	Computing all-pair shortest paths on four processors; <i>shaded areas</i> denote the portions of matrix $d(i, j)$ which must be stored within processor 3	171
Fig. 6.8	Speedup and execution time of the betweenness centrality algorithm in Boost, executed on the Intel cluster. The input graph (28,250 nodes, 692,668 edges) is replicated across the computing nodes. (a) Speedup. (b) Execution time	180
Fig. 6.9	Execution time for a Small-World graph with 28,250 nodes and 692,668 edges. The input graph is distributed across the computing nodes	181
Fig. 6.10	Speedup and execution time of the betweenness centrality algorithm in SNAP on the IBM p575, for a Small-World graph with 28,250 nodes and 692,668 edges. (a) Speedup. (b) Execution time	182
Fig. 6.11	Speedup and execution time of the betweenness centrality algorithm in SNAP on the IBM p575, for a larger Small-World graph with 224,288 nodes and 3 million edges. (a) Speedup. (b) Execution time	183
Fig. 6.12	Execution time of the betweenness centrality algorithm provided by the PBGL and by SNAP. Both algorithms have been run on the commodity cluster; SNAP has been executed on a single server using both CPU cores.....	184
Fig. 7.1	The processing pipeline for visual analysis of text combines data-intensive tasks (<i>top</i>) and user-centric tasks (<i>bottom</i>). <i>Solid black lines</i> indicate data flows while <i>dashed red lines</i> indicate user feedback to adapt automatic processes	191
Fig. 7.2	Semantic enrichment steps starting at single artifacts, e.g., documents, news articles, patents (<i>left</i>), and resulting in enriched representations in an index (<i>center</i>).....	193
Fig. 7.3	A visualization showing a search result set as a combination of tag clouds. Each polygonal area corresponds to a category of the documents in the search result set. Displayed named entities are enhanced with symbols indicating their type (person, location, data)	200

Fig. 7.4	An information landscape showing approx. 6,000 news articles on “computer industry” is used for drilling down to documents of interest: beginning with an overview (<i>left</i>) the user narrows down using topical cluster labels (<i>right</i>)	201
Fig. 7.5	Multidimensional visualization for books. <i>Left</i> : Scatterplot visualizing publication year (<i>x</i> -axis), page count (<i>y</i> -axis), file size (icon size), author (icon type); <i>right</i> : parallel coordinates showing nine metadata types on parallel axes	202
Fig. 7.6	Geo-visualization of Austria showing geo-references in news articles (<i>cones</i>). The size of the cone corresponds to the number of news articles for the particular geo-reference	204
Fig. 7.7	A stream visualization of approx. 750 news documents on “oil spill”, showing temporal development. Different gray values correspond to different topics.....	205
Fig. 7.8	A graph visualization of relationships between concepts extracted from a text data set (data courtesy of German National Library of Economics, 2011). Note that edge bundling is used to improve clarity and reduce clutter in the edge layout.....	206
Fig. 7.9	Examples for visually enhanced feedback. <i>Left</i> : Search results (<i>circles</i>) for comparing the topic overlap of different search engines (colors). Results with similar content are close. Sources can be interactively added or removed. <i>Right</i> : Visualizing classification decisions. Classes are arranged in a circle, data points are placed inside the circle according to their a-posteriori probabilities. Decisions can be corrected by drag and drop, classifier is retrained	209
Fig. 7.10	A coordinated multiple views GUI showing 6,900 news documents on “space”. Document selection (by time: from June to August), document coloring (each topical cluster in different color) and navigation in the hierarchy (location: Cluster 3 “shuttle, mars, columbia”) are coordinated	210
Fig. 7.11	A visualization of occurrences of Austrian politicians in search results. A rendered model of the parliament is used as visual metaphor. The figures of politicians are colored in their party color and scaled relative to the occurrence count. Clicking on a figure narrows the search result to articles containing the selected politician.....	212

Fig. 7.12	A visualization of co-occurrences of Austrian politicians in recent news media. Politicians are displayed as nodes connected by links representing co-occurrence strength by line width. Links are bundled to reveal high-level edge patterns. The strongest link visible is between the chancellor and the vice-chancellor	213
Fig. 7.13	The “Knowledge Space” visualization displaying the context of the encyclopedia entry for the mountaineer Reinhold Messner (<i>center</i>). The disc is divided into segments representing topics (e.g., “society” in the front). Related articles are represented by objects placed on the disc; shape, size and color encode additional metadata. For example, in the leftmost segment a geographic article (<i>circle</i>) and a premium content article (<i>diamond</i>) about the Mountain Everest is shown ..	214
Fig. 8.1	Private SVD with P4P.....	238
Fig. 8.2	Runtime ratios between homomorphic encryption based solutions and P4P	247

List of Tables

Table 2.1	A comparison of the discussed approaches for optimization of massively parallel data flows	70
Table 3.1	Graphs used (M: million. K: thousand)	77
Table 3.2	GIM-V in terms of SQL	86
Table 3.3	Parallelization choices. The last column of the table indicates whether the operation is parallelized in HEIGEN. Some operations are better to be run in parallel, since the input size is very large, while others are better in a single machine, since the input size is small and the overhead of parallel execution overshadows its decreased running time	95
Table 4.1	Example of a user profile with top-4 categories	119
Table 4.2	Top-10 categories along with their ODP precision and recall values	121
Table 4.3	Increase in runtime with respect to different aggregation levels	123
Table 5.1	Datasets	149
Table 5.2	SVM experiments	150
Table 5.3	Performance results for SVM training	150
Table 6.1	Summary of parallel graph libraries	174
Table 6.2	Technical specifications of the machines used for the tests	178
Table 7.1	Levels of and techniques used for semantic integration	194
Table 8.1	Performance comparison of existing MPC implementations	228
Table 8.2	Characteristics of the datasets	245
Table 8.3	Round complexity and precision	245
Table 8.4	SVD of large matrices	248

Acronyms

AHP	Analytical hierarchical process
APA	Austrian Press Agency
API	Application Programmer Interface
APP	Mobile application
AQL	Asterix query language
BFS	Breadth-First Search
BGL	Boost Graph Library
BOW	Bag of Words
C2S	Client to server
CC	Connected components
CDR	Call data record
CMV	Coordinated multiple views
DAG	Directed acyclic Graph
DBMS	Database Management System
DC	Disconnected component
DDL	Data definition language
DFS	Depth-First Search/Distributed File System
DML	Data manipulation language
EC2	Amazon Elastic Compute Cloud
ECC	Elliptic curve cryptography
EDR	Event data/detail record
EFF	Electronic Frontier Foundation
EM	Expectation maximization
FIFO	First In First Out
GCC	Giant connected component
GFS	Google File System
GIM-V	Generalized iterative matrix-vector multiplication
GPU	Graphics processing unit
HDFS	Hadoop Distributed File System
IAAS	Infrastructure As A Service
IE	Information extraction

IRAM	Implicitly Restarted Arnoldi Method
JSON	JavaScript object notation
KD	Knowledge discovery
LAN	Local area network
LINQ	Language INtegrated Query
LSI	Latent semantic indexing
MDS	Multidimensional scaling
MPC	Secure multi-party computation
MPI	Message Passing Interface
MPP	Massively parallel processing/processor
MPPN	Massively parallel processing node
MRF	MapReduce framework
MSA	Massive scale analytics
MST	Minimum spanning tree
MTGL	Multi-Threaded Graph Library
NAS	Network attached storage
NUMA	Non-uniform memory access
ODP	Open Directory Project
PACT	Parallelization contract
PBGL	Parallel Boost Graph Library
PCA	Principal component analysis
POS	Part of speech
PPDM	Privacy preserving data mining
PRG	Pseudo-random number generator
QP	Quadratic programming
RAID	Redundant array of independent/inexpensive disks
RAIN	Redundant array of independent/inexpensive nodes
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RRS	Recursive random search
RWR	Random Walk with Restart
S2S	Server to Server
SAN	Storage area network
SCC	Strongly connected components
SCOPE	Structured Computations Optimized for Parallel Execution
SMO	Sequential minimal optimization
SMP	Symmetric multi-processing machine
SNA	Social network analysis
SNAP	Small-world Network Analysis and Partitioning
SNS	Social network site
SQL	Structured query language
SSSP	Single source shortest path
SVD	Singular value decomposition
SVM	Support vector machine
TCP	Transmission Control Protocol

UDF	User-defined function
UMA	Uniform memory access
URL	Uniform resource locator
VSS	Verifiable secret sharing
ZKP	Zero-knowledge proof