Fundamentals of Stream Processing

Stream processing is a distributed computing paradigm that supports the gathering, processing, and analysis of high-volume, heterogeneous, continuous data streams to extract insights and actionable results in real time. This comprehensive, hands-on guide, combining the fundamental building blocks and emerging research in stream processing is ideal for application designers, system builders, analytic developers, as well as for students and researchers in the field. This book introduces the key components of the stream processing computing paradigm, including the distributed system infrastructure, the programming model, design patterns, and streaming analytics. The explanation of the underlying theoretical principles, illustrative examples, and implementations using the IBM InfoSphere Streams SPL language and real-world case studies provide students and practitioners with a comprehensive understanding of stream processing applications and the middleware that supports them.

Henrique C. M. Andrade is a vice president at JP Morgan and an adjunct associate professor in the Electrical Engineering Department at Columbia University. Along with Dr. Gedik, he is the co-inventor of the SPADE and the SPL stream processing languages. He has published over 50 peer-reviewed articles and is the co-recipient of the ACM SoftVis 2009, IEEE DSN 2011, and ACM DEBS 2011 best paper awards.

Buğra Gedik is in the faculty of the Computer Engineering Department, Bilkent University, Turkey. He is the co-inventor of the SPADE and the SPL stream processing languages. He has published over 50 peer-reviewed articles and is the co-recipient of the IEEE ICDCS 2003, IEEE DSN 2011, ACM DEBS 2011 and 2012, and IEEE ICWS 2013 best paper awards. He has been an Associate Editor for the *IEEE Transactions on Services Computing*. He has filed over 30 patents. He was named an IBM Master Inventor and is the recipient of an IBM Corporate Award.

Deepak S. Turaga is the manager of the Exploratory Stream Analytics department at the IBM T. J. Watson Research Center in Yorktown Heights and an adjunct associate professor in the Electrical Engineering Department at Columbia University. He has published over 75 peer reviewed articles, and has received the 2006 IEEE TCSVT best paper, and 2008 IEEE ICASSP best student paper awards. He has been an Associate Editor for the *IEEE Transactions CSVT* as well as *IEEE Transactions Multimedia*.

"This is the first comprehensive text on stream processing, covering details of stream analytic algorithms, programming language and application design, and finally systems issues. The use of several illustrative examples and real-world scenarios, coupled with advanced research topics, makes it very well suited for undergraduate and graduate classes."

Shih-Fu Chang, Columbia University

"In a world flooded with information, yet hungry for wisdom, you would find this refreshing and thorough treatment of stream computing an excellent resource for building systems that need to analyze live data to derive actionable insights."

Hans-Arno Jacobsen, University of Toronto

"A comprehensive guide to the field of stream processing covering a wide spectrum of analytical patterns against a specialized architecture for continuous processing. This reference will prove invaluable to those engaging in the fascinating field of continuous analysis. I wish it had been written when I started in this field!"

George Long, Senior System Architect

"This book is an excellent guide for anyone involved with stream processing or data-in-motion analytics in general and is a must-read for those using the InfoSphere Streams platform."

Jim Sharpe, President of Sharpe Engineering Inc.

"This book provides a very timely introduction to stream processing for engineers, students, and researchers. With the advent of Big Data, there is pressing need for real-time systems, algorithms, and languages for distributed streaming analysis. This book provides a comprehensive overview of the topic and is great for course work and also as a practitioner guide."

Mihaela van der Schaar, University of California, Los Angeles

"This is a first-of-its-kind book that takes a holistic approach to introduce stream processing – a technology that can help overcome the data deluge. The authors guide you through various system-level and analytical techniques for harnessing data-in-motion, using a clear exposition, supplemented with real-world scenarios. You will find this book an invaluable companion, whether you are an application developer, system builder, or an analytical expert."

Philip S. Yu, University of Illinois at Chicago

Fundamentals of Stream Processing

Application Design, Systems, and Analytics

HENRIQUE C. M. ANDRADE

JP Morgan, New York

BUĞRA GEDIK

Bilkent University, Turkey

DEEPAK S. TURAGA

IBM Thomas J. Watson Research Center, New York

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Published in the United States of America by Cambridge University Press, New York

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781107015548

First published 2014

Printed in the United Kingdom by Clays, St Ives plc

A catalogue record for this publication is available from the British Library

ISBN 978-1-107-01554-8 Hardback

Henrique dedicates this book to his parents, Gercil and Maria José, and to Kristen.

Buğra dedicates this book to his father Yusuf and to his mother Meral, and to all his teachers and mentors through life.

Deepak dedicates this book to his parents Sudha and Ravi, his wife Swapna, and daughter Mythri.

Contents

Preface

Stream processing is a paradigm built to support natural and intuitive ways of designing, expressing, and implementing *continuous* online high-speed data processing. If we look at systems that manage the critical infrastructure that makes modern life possible, each of their components must be able to *sense* what is happening externally, by processing continuous inputs, and to *respond* by continuously producing results and actions. This pattern is very intuitive and is not very dissimilar from how the human body works, constantly sensing and responding to external stimuli. For this reason, stream processing is a natural way to analyze information as well as to interconnect the different components that make such processing fast and scalable.

We wrote this book as a comprehensive reference for students, developers, and researchers to allow them to design and implement their applications using the stream processing paradigm. In many domains, employing this paradigm yields results that better match the needs of certain types of applications, primarily along three dimensions.

First, many applications naturally adhere to a sense-and-respond pattern. Hence, engineering these types of applications is simpler, as both the programming model and the supporting stream processing systems provide abstractions and constructs that match the needs associated with continuously sensing, processing, predicting, and reacting.

Second, the stream processing paradigm naturally supports extensibility and scalability requirements. This allows stream processing applications to better cope with high data volumes, handle fluctuations in the workload and resources, and also readjust to time-varying data and processing characteristics.

Third, stream processing supports the use of new algorithmic and analytical techniques for *online* mining of both structured data (such as relational database-style records) as well as unstructured data (such as audio, video, text, and image). This breaks the cycle of storing the incoming data first to analyze it later, and makes it possible to considerably shorten the lag between sensing and responding.

After more than a decade of research in this space, stream processing has had a prolific and successful history in academic and industrial settings. Several advances in data analysis and management, signal processing, data mining, optimization theory, as well as in distributed systems technology, have provided a strong foundation for the development of research and commercial stream processing systems. In essence, stream processing is no longer an emerging paradigm, it is now ready for prime time.

The stream processing paradigm can now be harnessed in at least two ways. First, it can be used to transition existing legacy applications into true streaming implementations, making them more flexible, scalable, and adaptive. Second, stream processing can also be used to implement new analytics-intensive, high-performance, and innovative applications that could not be practically engineered earlier. Indeed, as will be seen in this book, stream processing applications can now be elegantly designed to be adaptive and autonomic, as well as self-evolving and able to continuously make use of newly learned knowledge.

Considering all of these aspects, this book is designed to provide a comprehensive foundation on stream processing techniques and on the skills necessary to design and develop stream processing applications. The book is divided into five major parts.

In Part I, we start with a discussion on the trends that led to development of the stream processing paradigm, providing also an overview of the initial academic efforts on analytical techniques, and on the engineering of some of the early stream processing system prototypes.

In Part II, we focus on application development. We describe core concepts of stream processing application development and illustrate them using the SPL language. SPL is the dataflow programming language provided by InfoSphere Streams, a commercial distributed stream processing system.

In Part III, we shift our attention to the architecture of stream processing systems. We first describe a conceptual middleware software architecture and its required services to support efficient, scalable, and fault-tolerant stream processing applications. We then illustrate these concepts with the architectural organization of InfoSphere Streams, shedding light on its internal components, and on the application runtime environment it exposes to a developer.

In Part IV, we build on the foundation provided in the earlier two parts to discuss how to best structure and design a stream processing application. The focus in this part of the book is on design patterns and principles common to stream processing applications, as well as on the algorithms used to implement online analytics.

In Part V, we describe a few case studies, detailing the end-to-end process of designing, implementing, and refining stream processing applications. This part brings together all of the information distilled in the earlier parts of the book. The case studies include real-world applications and showcase typical design decisions that must be made by developers.

We have designed this book to be used for undergraduate- as well as graduate-level courses on stream processing. The book's content is also structured so that application developers and system analysts can quickly develop the skills needed to make use of the stream processing paradigm.

While there are many ways to structure a semester-long course on stream processing, we recommend the following breakdown:

- For an undergraduate course, we believe that substantial focus should be devoted to the algorithmic and application-building aspects of stream processing. We believe that the majority of the time should be spent in Part II, where the focus should be on

teaching the SPL programming language. This training will provide a solid foundation for tackling Part IV, where the algorithmic and analytical techniques are discussed. This hands-on portion of the class should be concluded with the contents of Part V, to discuss case studies that show how a complete application might be designed. Information from Part III, particularly from Chapter 8, can be used as needed to provide basic system administration knowledge on managing the Streams system. We stress that a stream processing undergraduate-level course must be hands-on, so that students can pick up important technical skills along the way. At the end of some chapters, we suggest exercises that can be used to solidify a working knowledge of the SPL language and the Streams platform. Finally, we think the class should culminate with a medium size final project of the magnitude of the case studies described in Part V.

- For a graduate course, we believe that the emphasis should be on the theoretical foundations and research issues surrounding the algorithmic, analytical, software engineering, and distributed processing architectural foundations of stream processing. Nevertheless, a semester-long course should also provide a solid programming foundation and an understanding of the practical aspects of building stream processing applications. Hence, our suggestion is to follow a similar organization as for an undergraduate course on a compressed time scale, but augmented with selected readings from the bibliography included at the end of each chapter. For this extra reading, we suggest focusing primarily on a subset of the foundational papers listed in Chapters 1, 2, 10, and 11. Despite its maturity, stream processing is still a very fertile area of research. We offer suggestions of possible research topics and open problems in Section 13.2. These areas can be used for individual short-term research projects, as well as for more advanced studies leading to a thesis. One final suggestion we can offer is to make use of the case studies, discussed in Part V, as the motivation for the course's final research projects. These projects can tackle one or more of the supporting analytical- or distributed system-related parts of stream processing, particularly where the state-of-the-art can be further advanced.

We believe that this book is self-contained and no specific formal background is necessary. Yet a reader might benefit from prior knowledge of a modern programming language such as Java and C++, as well as experience with scripting languages such as Perl and Python. Likewise, previous experience with relational databases, data mining platforms, optimization theory, and signal processing can also be directly leveraged. Content that is complementary to this book covering several of the practical aspects of using Streams, including its set of documentation manuals and the Streams' IBM RedBook, are linked from `www.thestreamprocessingbook.info/ibm/streams-infocenter` and `www.thestreamprocessingbook.info/ibm/streams-redbook`, respectively.

We think that the best way to learn how to use a new technology is by trying it out and we include several code excerpts and examples of how to use InfoSphere Streams, to illustrate the fundamental concepts appropriately. For readers interested in obtaining

an InfoSphere Streams license, commercial users can contact IBM directly as well as any of its authorized resellers. IBM also provides time-limited trial licenses for InfoSphere Streams. Additional information and specific conditions on the trial program can be found at `www.thestreamprocessingbook.info/ibm/streams-main`. IBM also maintains a program that enables academic users to obtain a license free of charge. This type of license can be used in a teaching environment. Additional information and specific conditions on the IBM academic program can be found at `www.thestreamprocessingbook.info/ibm/academic-initiative`.

As we mentioned, InfoSphere Streams and its SPL language are used throughout this book as examples of a stream processing system and programming language. In this way, we can provide a conceptual overview, coupled with practical foundations and code examples, application design challenges, and finally system administration issues. We believe that the abstractions, concepts, and examples included in the book are general enough to be used with a different stream processing system, both in a teaching or commercial setting.

As a commercial product, Streams is evolving and, periodically, new versions will become available. In this book, we have made an effort to provide working code and other usage examples consistent with version 3.0, the latest available version as of November 2012.

We hope that the reader will find this book as exciting to read as it was to write. We have attempted to balance the content such that it is useful both to readers who are attempting to familiarize themselves with the stream processing paradigm, and for advanced readers who intend to develop new stream processing applications, systems, and algorithms.

Finally, we welcome feedback on this book as well as accounts of experiences using stream processing in academic and industrial settings. The authors can be contacted through the website that accompanies this book at `www.thestreamprocessingbook.info`, where readers will also find a code repository with example applications and code excerpts (`www.thestreamprocessingbook.info/apps`) as well as this book's errata (`www.thestreamprocessingbook.info/errata`).

Foreword

Humans are deeply curious and expend boundless energy and thought in sensing and interpreting that which surrounds them. Over time, direct perception through the five physical senses was extended by the creation of ingenious instrumentation designed to magnify and amplify weak signals, bringing what was beyond the visible into focus. Telescopes and light microscopy revealed natural phenomena that enabled richer and more sophisticated theories and understanding.

In recent decades industrialization has filled the world with machines and complex systems that manufacture, transport, track and deliver, communicate and mediate financial and social transactions, entertain and educate, treat and repair, and perform thousands of other tasks. As was true with the natural world, human curiosity seeks to understand the operation of these machines and systems and their interactions, but now with the added urgency to understand how and how well systems are operating, and often why they are not working as intended. Direct perception is no longer effective, nor is observation through the mere amplification of our five senses. Specialized sensors capture phenomena such as vibration, frequency, complex movements, or human-generated data and messages produced by the machines and systems as a side-effect of their operation, and so perception must now be through computer-aided interpretation of the digital signals.

Up until recently, no systematic approach existed for the creation of digital signal interpretation required to engineer this new class of perception mechanisms. IBM has long recognized this, and early in 2004 initiated a research program to create such a new approach from the ground up. As the lead for this program, I assembled a multi-disciplinary team of experts in distributed systems, mathematics, programming languages, machine learning and data mining, and computer science theory, and, over a five-year period, we (∼60 researchers and engineers) developed *System S* – the research precursor to the IBM InfoSphere Streams product described in this volume, to illustrate stream processing concepts and techniques. During this period, the stream processing model and its underlying system structures evolved through feedback and application of the technology in a wide variety of real-world contexts.

The authors of this book were central members of this research and development effort, and have been closely involved in all aspects of the program – from conceptualization of the objectives, to the design of the architecture and the programming model, to engineering and implementing the system, and finally designing analytic applications and deploying them in real-world settings. Each of the three authors focused their work

on a core area: the weaving of analytics into sophisticated applications, the language in which applications are described, and the system and runtime that support execution of the applications. This provided the authors with a complete and unique insider's perspective on both the fundamentals of this new model and realizations of the model in practical implementations and deployments.

This magnificent volume has captured this knowledge for students, researchers, and practitioners. The book provides an in-depth introduction to the stream processing paradigm, its programming model, its distributed runtime, and its analytic applications that will enable readers to use these techniques effectively in various settings for complex environmental monitoring and control applications. The book also includes several sample implementations, algorithms, and design principles, along with real-world use cases to provide hands-on training to practitioners. Finally, the book provides advanced readers and researchers with the necessary fundamentals to enable the design and extension of the stream processing paradigm to solve future problems.

All of us involved with the creation of this technology are convinced that stream processing will become a permanent element in the quest to create ever more sophisticated instrumentation to better understand our world, the machines and systems that serve us, and the interaction among them. The impact will be felt in virtually all modern industrial sectors and its use will lead to a safer, more efficient, and more productive society.

Nagui Halim
IBM Fellow
IBM T. J. Watson Research Center
Yorktown Heights, NY
United States

Acknowledgements

Henrique, Buğra, and Deepak were colleagues at IBM Research for several years, and were part of the team that designed and implemented InfoSphere Streams. This team included researchers, architects, designers, project managers, programmers, and testers under the direction of Nagui Halim. The techniques and lessons described here are the product of numerous discussions and refinements with this much larger team. Therefore, we acknowledge the collaboration of colleagues from multiple IBM sites, including Rochester (MN), Toronto (Canada), Silicon Valley (CA), Raleigh (NC), the China Research Lab, and from the Thomas J. Watson Research Center (NY). This book is also a tribute to our talented IBM colleagues. Thank you!

It really takes a village to build a system such as InfoSphere Streams, as we ourselves learned in the process. So it came as no surprise that the same applies when writing a book. We owe a debt of gratitude to the expertise of our colleagues who also helped reviewing an early version of this manuscript. We are particularly thankful to these early "settlers" who braved the elements to provide us with many helpful suggestions and corrections: Tarık Arıcı (İstanbul Şehir Üniversitesi), John Cox (US Government), Renato A. Ferreira (Universidade Federal de Minas Gerais), Andy Frenkiel (IBM Research), Martin Hirzel (IBM Research), Gabriela Jacques da Silva (IBM Research), Paul Jones (HM Government), Rohit Khandekar (Knight Capital Group), Senthil Nathan (IBM Research), Scott Schneider (IBM Research), Robert Soulé (Cornell University), William Szewczyk (US Government), Rohit Wagle (IBM Research), Brian Williams (IBM Software Services Federal), and Kun-Lung Wu (IBM Research).

A special thanks is due to our colleague Wim De Pauw (IBM Research) who graciously produced the image we use on this book's cover, a depiction of a stream processing application.

Acronyms

A/D Analog-to-Digital
AAS Authentication and Authorization Service
ACID Atomicity, Consistency, Isolation, and Durability
ACL Access Control List
ADL Application Description Language
ADT Abstract Data Type
AES Advanced Encryption Standard
AIS Agrawal, Imielinski, Swami
AMS Alon, Matias, and Szegedy
ANSI American National Standards Institute
API Application Programming Interface
ARIMA Auto Regressive Integrated Moving Average
ASCII American Standard Code for Information Interchange
ATE Automated Test Equipment
BIH Beth Israel Hospital
BI Business Intelligence
BJKST Bar-Yossef, Jayram, Kumar, Sivakumar, Trevisan
CART Classification And Regression Tree
CC Command and Control
CDR Call Detail Record
CEP Complex Event Processing
CIS Clinical Information System
CKRM Class-based Kernel Resource Management
CORBA Common Object Request Broker Architecture
CPU Central Processing Unit
CQL Continuous Query Language
CQ Continuous Query
CSV Comma-Separated Value
CVFDT Concept-adapting Very Fast Decision Tree learner
DAG Directed Acyclic Graph
DBMS Data Base Management System
DBSCAN Density-Based Spatial Clustering of Applications with Noise
DCOM Distributed Component Object Model
DCT Discrete Cosine Transform

DDL Data Definition Language
DDoS Distributed Denial of Service
DFT Discrete Fourier Transform
DHT Distributed Hash Table
DMG Data Mining Group
DML Data Manipulation Language
DNS Domain Name System
DOM Document Object Model
DoS Denial of Service
DPI Deep Packet Inspection
DSL Domain-Specific Language
DSO Dynamically Shared Object
DSS Decision Support System
DTD Document Type Definition
ECA Event-Condition-Action
ECG Electrocardiogram
EDA Electronic Design Automation
EEG Electroencephalogram
EKG Elektrokardiogramm
EM Expectation Maximization
EMS Emergency Medical Services
EPL Event Processing Language
EPN Event Processing Network
EPFL École Polytechnique Fédérale de Lausanne
ER Entity Relationship
ESP Event Stream Processor
ETL Extract/Transform/Load
FDC Fault Detection and Classification
FFT Fast Fourier Transform
FPGA Field-Programmable Gate Array
FSF Free Software Foundation
FTP File Transfer Protocol
Gbps gigabits per second
GMM Gaussian Mixture Model
GPS Global Positioning System
GPU Graphics Processing Unit
GRAM Globus Resource Allocation Manager
GSM Global System for Mobile Communications
GSN Global Sensor Networks
GSQL Gigascope SQL
GUI Graphical User Interface
HA High Availability
HC Host Controller
HDFS Hadoop Distributed File System

HMM Hidden Markov Model
HTML HyperText Markup Language
HTTP HyperText Transfer Protocol
HTTPS HyperText Transfer Protocol Secure
Hz hertz
I/O Input/Output
ICU Intensive Care Unit
IDDQ Direct Drain Quiescent Current
IDE Integrated Development Environment
IDL Interface Definition Language
IDS Intrusion Detection System
IEEE Institute of Electrical and Electronics Engineers
IOR Interoperable Object Reference
IP Internet Protocol
IPC Inter-Process Communication
IT Information Technology
JDBC Java Data Base Connectivity
JMS Java Message Service
JNI Java Native Interface
JSON JavaScript Object Notation
JVM Java Virtual Machine
kbps kilobits per second
KHz kilohertz
KLT Karhunen–Loève Transform
KNN k-Nearest Neighbors
LAN Local Area Network
LBG Linde–Buzo–Gray
LDAP Lightweight Directory Access Protocol
LDA Linear Discriminant Analysis
LFUP Least Frequently Updated Partition
LLM Low Latency Messaging
LF Line Fit
LPC Linear Predictive Coding
LRUP Least Recently Updated Partition
MIMD Multiple Instruction Multiple Data
MIT Massachusetts Institute of Technology
MLA Manifold Learning Algorithm
MLE Maximum Likelihood Estimation
MLP Multi-Layer Perception
MPI Message Passing Interface
MPQ Moment Preserving Quantization
MP megapixel
ms millisecond
MTDF Mean Time to Detect Failures

MUSCLES MUlti-SequenCe LEast Squares
mV millivolt
NB Naïve Bayes
NFS Network File System
NICU Neonatal Intensive Care Unit
NN Nearest Neighbors
NOAA National Oceanic and Atmospheric Administration
NP Non-deterministic Polynomial time
NPMR Non-Parametric Multiplicative Regression
NS Name Service
NYSE New York Stock Exchange
ODBC Open Data Base Connectivity
ODBMS Object Data Base Management System
OLAP Online Analytical Processing
OLTP Online Transaction Processing
OMG Object Management Group
OM Operations Monitoring
OO Object-Oriented
OP Oldest Partition
OS Operating System
PAM Pluggable Authentication Module
PCA Principal Component Analysis
PCR Parent–Child Relationship
PDF Probability Density Function
PDMS Patient Data Management System
PEC Processing Element Container
PE Processing Element
PIPES Public Infrastructure for Processing and Exploring Streams
PLY Performance Limited Yield
PMF Probability Mass Function
PMML Predictive Model Markup Language
PM Patient Monitoring
POJO Plain Old Java Object
PSRO Performance Sort Ring Oscillator
PVM Parallel Virtual Machine
QoS Quality of Service
RAD Rapid Application Development
RBF Radial Basis Function
RDF Random Decision Forest
RDMA Remote Direct Memory Access
RFID Radio-Frequency IDentification
ROC Receiver Operating Characteristic
RPC Remote Procedure Call
RSS Really Simple Syndication

RTP Real-time Transport Protocol
SAM Streams Application Manager
SAN Storage Area Network
SAX Simple API for XML
SCADA Supervisory Control and Data Acquisition
SCH Scheduler
SDE Semi-Definite Embedding
SGD Stochastic Gradient Descent
SIMD Single Instruction Multiple Data
SLR Single Logistic Regression
SMS Short Message System
SMTP Simple Mail Transfer Protocol
SOAP Simple Object Access Protocol
SOM Self-Organizing Map
SPA Stream Processing Application
SPADE Stream Processing Application Declarative Engine
SPC Semiconductor Process Control
SPIRIT Streaming Pattern dIscoveRy in multIple Timeseries
SPS Stream Processing System
SQL Structured Query Language
SQuAl Stream Query Algebra
SRM Streams Resource Manager
SSH Secure SHell
SSL Secure Sockets Layer
SVD Singular Value Decomposition
SVM Support Vector Machine
SWS Streams Web Server
TB terabytes
TCP Transmission Control Protocol
TEDS Transducer Electrical Data Sheet
TelegraphCQ Telegraph Continuous Queries
UCLA University of California, Los Angeles
UDP User Datagram Protocol
UML Unified Modeling Language
URI Uniform Resource Identifier
UX User Experience
VFDT Very Fast Decision Tree
VFML Very Fast Machine Learning
VLSI Very Large Scale Integration
VoIP Voice over IP
VWAP Volume Weighted Average Price
WAN Wide Area Network
WSDL Web Services Description Language
WTTW Who is Talking To Whom

XHTML eXtensible HyperText Markup Language
XML eXtensible Markup Language
XPATH XML Path Language
XSD XML Schema Definition
XSLT eXtensible Stylesheet Language Transformations
ZIP Zone Improvement Plan