contents

preface xiii
acknowledgments xv
about this book xviii

1 *A new paradigm for Big Data 1*

1.1 How this book is structured 2
1.2 Scaling with a traditional database 3
Scaling with a queue 3 ▪ Scaling by sharding the database 4 Fault-tolerance issues begin 5 ▪ Corruption issues 5 ▪ What went wrong? 5 ▪ How will Big Data techniques help? 6
1.3 NoSQL is not a panacea 6
1.4 First principles 6
1.5 Desired properties of a Big Data system 7
Robustness and fault tolerance 7 ▪ Low latency reads and updates 8 ▪ Scalability 8 ▪ Generalization 8 ▪ Extensibility 8 Ad hoc queries 8 ▪ Minimal maintenance 9 ▪ Debuggability 9
1.6 The problems with fully incremental architectures 9
Operational complexity 10 ▪ Extreme complexity of achieving eventual consistency 11 ▪ Lack of human-fault tolerance 12 Fully incremental solution vs. Lambda Architecture solution 13

1.7 Lambda Architecture 14

Batch layer 16 ▪ Serving layer 17 ▪ Batch and serving layers satisfy almost all properties 17 ▪ Speed layer 18

1.8 Recent trends in technology 20

CPUs aren't getting faster 20 ▪ Elastic clouds 21 ▪ Vibrant open source ecosystem for Big Data 21

1.9 Example application: SuperWebAnalytics.com 22

1.10 Summary 23

PART 1 BATCH LAYER 25

2 *Data model for Big Data 27*

2.1 The properties of data 29

Data is raw 31 ▪ Data is immutable 34 ▪ Data is eternally true 36

2.2 The fact-based model for representing data 37

Example facts and their properties 37 ▪ Benefits of the fact-based model 39

2.3 Graph schemas 43

Elements of a graph schema 43 ▪ The need for an enforceable schema 44

2.4 A complete data model for SuperWebAnalytics.com 45

2.5 Summary 46

3 *Data model for Big Data: Illustration 47*

3.1 Why a serialization framework? 48

3.2 Apache Thrift 48

Nodes 49 ▪ Edges 49 ▪ Properties 50 ▪ Tying everything together into data objects 51 ▪ Evolving your schema 51

3.3 Limitations of serialization frameworks 52

3.4 Summary 53

4 *Data storage on the batch layer 54*

4.1 Storage requirements for the master dataset 55

4.2 Choosing a storage solution for the batch layer 56

Using a key/value store for the master dataset 56 ▪ Distributed filesystems 57

4.3 How distributed filesystems work 58

4.4 Storing a master dataset with a distributed filesystem 59

4.5 Vertical partitioning 61

4.6 Low-level nature of distributed filesystems 62

4.7 Storing the SuperWebAnalytics.com master dataset on a distributed filesystem 64

4.8 Summary 64

5 ***Data storage on the batch layer: Illustration 65***

5.1 Using the Hadoop Distributed File System 66

The small-files problem 67 ▪ *Towards a higher-level abstraction 67*

5.2 Data storage in the batch layer with Pail 68

Basic Pail operations 69 ▪ *Serializing objects into pails 70*
Batch operations using Pail 72 ▪ *Vertical partitioning with Pail 73* ▪ *Pail file formats and compression 74* ▪ *Summarizing the benefits of Pail 75*

5.3 Storing the master dataset for SuperWebAnalytics.com 76

A structured pail for Thrift objects 77 ▪ *A basic pail for SuperWebAnalytics.com 78* ▪ *A split pail to vertically partition the dataset 78*

5.4 Summary 82

6 ***Batch layer 83***

6.1 Motivating examples 84

Number of pageviews over time 84 ▪ *Gender inference 85*
Influence score 85

6.2 Computing on the batch layer 86

6.3 Recomputation algorithms vs. incremental algorithms 88

Performance 89 ▪ *Human-fault tolerance 90* ▪ *Generality of the algorithms 91* ▪ *Choosing a style of algorithm 91*

6.4 Scalability in the batch layer 92

6.5 MapReduce: a paradigm for Big Data computing 93

Scalability 94 ▪ *Fault-tolerance 96* ▪ *Generality of MapReduce 97*

6.6 Low-level nature of MapReduce 99

Multistep computations are unnatural 99 ▪ *Joins are very complicated to implement manually 99* ▪ *Logical and physical execution tightly coupled 101*

6.7 Pipe diagrams: a higher-level way of thinking about batch computation 102

Concepts of pipe diagrams 102 ▪ Executing pipe diagrams via MapReduce 106 ▪ Combiner aggregators 107 ▪ Pipe diagram examples 108

6.8 Summary 109

7 ***Batch layer: Illustration 111***

7.1 An illustrative example 112

7.2 Common pitfalls of data-processing tools 114

Custom languages 114 ▪ Poorly composable abstractions 115

7.3 An introduction to JCascalog 115

The JCascalog data model 116 ▪ The structure of a JCascalog query 117 ▪ Querying multiple datasets 119 ▪ Grouping and aggregators 121 ▪ Stepping though an example query 122 Custom predicate operations 125

7.4 Composition 130

Combining subqueries 130 ▪ Dynamically created subqueries 131 ▪ Predicate macros 134 ▪ Dynamically created predicate macros 136

7.5 Summary 138

8 ***An example batch layer: Architecture and algorithms 139***

8.1 Design of the SuperWebAnalytics.com batch layer 140

Supported queries 140 ▪ Batch views 141

8.2 Workflow overview 144

8.3 Ingesting new data 145

8.4 URL normalization 146

8.5 User-identifier normalization 146

8.6 Deduplicate pageviews 151

8.7 Computing batch views 151

Pageviews over time 151 ▪ Unique visitors over time 152 Bounce-rate analysis 152

8.8 Summary 154

9 *An example batch layer: Implementation* 156

9.1 Starting point 157
9.2 Preparing the workflow 158
9.3 Ingesting new data 158
9.4 URL normalization 162
9.5 User-identifier normalization 163
9.6 Deduplicate pageviews 168
9.7 Computing batch views 169
Pageviews over time 169 ▪ Uniques over time 171 ▪ Bounce-rate analysis 172
9.8 Summary 175

PART 2 SERVING LAYER....177

10 *Serving layer* 179

10.1 Performance metrics for the serving layer 181
10.2 The serving layer solution to the normalization/denormalization problem 183
10.3 Requirements for a serving layer database 185
10.4 Designing a serving layer for SuperWebAnalytics.com 186
Pageviews over time 186 ▪ Uniques over time 187 ▪ Bounce-rate analysis 188
10.5 Contrasting with a fully incremental solution 188
Fully incremental solution to uniques over time 188 ▪ Comparing to the Lambda Architecture solution 194
10.6 Summary 195

11 *Serving layer: Illustration* 196

11.1 Basics of ElephantDB 197
View creation in ElephantDB 197 ▪ View serving in ElephantDB 197 ▪ Using ElephantDB 198
11.2 Building the serving layer for SuperWebAnalytics.com 200
Pageviews over time 200 ▪ Uniques over time 202 ▪ Bounce-rate analysis 203
11.3 Summary 204

PART 3 SPEED LAYER 205

12 *Realtime views* 207

12.1 Computing realtime views 209

12.2 Storing realtime views 210

Eventual accuracy 211 ▪ Amount of state stored in the speed layer 211

12.3 Challenges of incremental computation 212

Validity of the CAP theorem 213 ▪ The complex interaction between the CAP theorem and incremental algorithms 214

12.4 Asynchronous versus synchronous updates 216

12.5 Expiring realtime views 217

12.6 Summary 219

13 *Realtime views: Illustration* 220

13.1 Cassandra's data model 220

13.2 Using Cassandra 222

Advanced Cassandra 224

13.3 Summary 224

14 *Queuing and stream processing* 225

14.1 Queuing 226

Single-consumer queue servers 226 ▪ Multi-consumer queues 228

14.2 Stream processing 229

Queues and workers 230 ▪ Queues-and-workers pitfalls 231

14.3 Higher-level, one-at-a-time stream processing 231

Storm model 232 ▪ Guaranteeing message processing 236

14.4 SuperWebAnalytics.com speed layer 238

Topology structure 240

14.5 Summary 241

15 *Queuing and stream processing: Illustration* 242

15.1 Defining topologies with Apache Storm 242

15.2 Apache Storm clusters and deployment 245

15.3 Guaranteeing message processing 247

15.4 Implementing the SuperWebAnalytics.com uniques-over-time speed layer 249

15.5 Summary 253

16 *Micro-batch stream processing* 254

16.1 Achieving exactly-once semantics 255

Strongly ordered processing 255 ▪ Micro-batch stream processing 256 ▪ Micro-batch processing topologies 257

16.2 Core concepts of micro-batch stream processing 259

16.3 Extending pipe diagrams for micro-batch processing 260

16.4 Finishing the speed layer for SuperWebAnalytics.com 262

Pageviews over time 262 ▪ Bounce-rate analysis 263

16.5 Another look at the bounce-rate-analysis example 267

16.6 Summary 268

17 *Micro-batch stream processing: Illustration* 269

17.1 Using Trident 270

17.2 Finishing the SuperWebAnalytics.com speed layer 273

Pageviews over time 273 ▪ Bounce-rate analysis 275

17.3 Fully fault-tolerant, in-memory, micro-batch processing 281

17.4 Summary 283

18 *Lambda Architecture in depth* 284

18.1 Defining data systems 285

18.2 Batch and serving layers 286

Incremental batch processing 286 ▪ Measuring and optimizing batch layer resource usage 293

18.3 Speed layer 297

18.4 Query layer 298

18.5 Summary 299

index 301