

brief contents

1 ■ Introducing Storm	1
2 ■ Core Storm concepts	12
3 ■ Topology design	33
4 ■ Creating robust topologies	76
5 ■ Moving from local to remote topologies	102
6 ■ Tuning in Storm	130
7 ■ Resource contention	161
8 ■ Storm internals	187
9 ■ Trident	207

1 *Introducing Storm 1*

1.1 What is big data?	2
<i>The four Vs of big data 2 • Big data tools</i>	3
1.2 How Storm fits into the big data picture	6
<i>Storm vs. the usual suspects</i>	8
1.3 Why you'd want to use Storm	10
1.4 Summary	11

2 *Core Storm concepts 12*

2.1 Problem definition: GitHub commit count dashboard	12
<i>Data: starting and ending points</i>	13
<i>Breaking down the problem</i>	14
2.2 Basic Storm concepts	14
<i>Topology 15 • Tuple 15 • Stream 18 • Spout 19</i>	
<i>Bolt 20 • Stream grouping</i>	22
2.3 Implementing a GitHub commit count dashboard in Storm	24
<i>Setting up a Storm project 25 • Implementing the spout</i>	25
<i>Implementing the bolts 28 • Wiring everything together to form the topology</i>	31
2.4 Summary	32

3 *Topology design 33*

3.1 Approaching topology design	34
3.2 Problem definition: a social heat map	34
<i>Formation of a conceptual solution</i>	35
3.3 Precepts for mapping the solution to Storm	35
<i>Consider the requirements imposed by the data stream</i>	36

<i>Represent data points as tuples</i>	37	<i>Steps for determining the topology composition</i>	38
3.4 Initial implementation of the design	40	<i>Spout: read data from a source</i>	41
		<i>Bolt: connect to an external service</i>	42
		<i>Bolt: collect data in-memory</i>	44
		<i>Bolt: persisting to a data store</i>	48
		<i>Defining stream groupings between the components</i>	51
		<i>Building a topology for running in local cluster mode</i>	51
3.5 Scaling the topology	52	<i>Understanding parallelism in Storm</i>	54
		<i>Adjusting the topology to address bottlenecks inherent within design</i>	58
		<i>Adjusting the topology to address bottlenecks inherent within a data stream</i>	64
3.6 Topology design paradigms	69	<i>Design by breakdown into functional components</i>	70
		<i>Design by breakdown into components at points of repartition</i>	71
		<i>Simplest functional components vs. lowest number of repartitions</i>	74
3.7 Summary	74		

4 Creating robust topologies 76

4.1 Requirements for reliability	76	<i>Pieces of the puzzle for supporting reliability</i>	77
4.2 Problem definition: a credit card authorization system	77	<i>A conceptual solution with retry characteristics</i>	78
		<i>Defining the data points</i>	79
		<i>Mapping the solution to Storm with retry characteristics</i>	80
4.3 Basic implementation of the bolts	81	<i>The AuthorizeCreditCard implementation</i>	82
		<i>The ProcessedOrderNotification implementation</i>	83
4.4 Guaranteed message processing	84	<i>Tuple states: fully processed vs. failed</i>	84
		<i>Anchoring, acking, and failing tuples in our bolts</i>	86
		<i>A spout's role in guaranteed message processing</i>	90
4.5 Replay semantics	94	<i>Degrees of reliability in Storm</i>	94
		<i>Examining exactly once processing in a Storm topology</i>	95
		<i>Examining the reliability guarantees in our topology</i>	95
4.6 Summary	101		

5 Moving from local to remote topologies 102

5.1 The Storm cluster	103	<i>The anatomy of a worker node</i>	104
		<i>Presenting a worker node within the context of the credit card authorization topology</i>	106
5.2 Fail-fast philosophy for fault tolerance within a Storm cluster	108		
5.3 Installing a Storm cluster	109	<i>Setting up a Zookeeper cluster</i>	109
		<i>Installing the required Storm dependencies to master and worker nodes</i>	110
		<i>Installing Storm to master and worker nodes</i>	110
		<i>Configuring the master and worker nodes via storm.yaml</i>	110
		<i>Launching Nimbus and</i>	

<i>Supervisors under supervision</i>	111
5.4 Getting your topology to run on a Storm cluster	112
<i>Revisiting how to put together the topology components</i>	112
<i>Running topologies in local mode 113 • Running topologies on a remote Storm cluster 114 • Deploying a topology to a remote Storm cluster</i>	114
5.5 The Storm UI and its role in the Storm cluster	116
<i>Storm UI: the Storm cluster summary 116 • Storm UI: individual Topology summary 120 • Storm UI: individual spout/bolt summary</i>	124
5.6 Summary	129

6 Tuning in Storm **130**

6.1 Problem definition: Daily Deals! reborn	131
<i>Formation of a conceptual solution 132 • Mapping the solution to Storm concepts</i>	132
6.2 Initial implementation	133
<i>Spout: read from a data source 134 • Bolt: find recommended sales 135 • Bolt: look up details for each sale</i>	136
<i>Bolt: save recommended sales</i>	138
6.3 Tuning: I wanna go fast	139
<i>The Storm UI: your go-to tool for tuning</i>	139
<i>Establishing a baseline set of performance numbers</i>	140
<i>Identifying bottlenecks 142 • Spouts: controlling the rate data flows into a topology</i>	145
6.4 Latency: when external systems take their time	148
<i>Simulating latency in your topology 148 • Extrinsic and intrinsic reasons for latency</i>	150
6.5 Storm's metrics-collecting API	154
<i>Using Storm's built-in CountMetric 154 • Setting up a metrics consumer 155 • Creating a custom SuccessRateMetric</i>	156
<i>Creating a custom MultiSuccessRateMetric</i>	158
6.6 Summary	160

7 Resource contention **161**

7.1 Changing the number of worker processes running on a worker node	163
<i>Problem 163 • Solution 164 • Discussion</i>	165
7.2 Changing the amount of memory allocated to worker processes (JVMs)	165
<i>Problem 165 • Solution 165 • Discussion</i>	166
7.3 Figuring out which worker nodes/processes a topology is executing on	166
<i>Problem 166 • Solution 166 • Discussion</i>	167
7.4 Contention for worker processes in a Storm cluster	168
<i>Problem 169 • Solution 170 • Discussion</i>	171
7.5 Memory contention within a worker process (JVM)	171
<i>Problem 174 • Solution 174 • Discussion</i>	175
7.6 Memory contention on a worker node	175

<i>Problem 178 . Solution 178 . Discussion</i>	178
7.7 Worker node CPU contention	178
<i>Problem 179 . Solution 179 . Discussion</i>	181
7.8 Worker node I/O contention	181
<i>Network/socket I/O contention 182 . Disk I/O contention</i>	184
7.9 Summary	186

8 *Storm internals* **187**

8.1 The commit count topology revisited	188
<i>Reviewing the topology design 188 . Thinking of the topology as running on a remote Storm cluster 189 . How data flows between the spout and bolts in the cluster</i>	189
8.2 Diving into the details of an executor	191
<i>Executor details for the commit feed listener spout</i>	191
<i>Transferring tuples between two executors on the same JVM</i>	192
<i>Executor details for the email extractor bolt 194 . Transferring tuples between two executors on different JVMs 195 . Executor details for the email counter bolt</i>	197
8.3 Routing and tasks	198
8.4 Knowing when Storm's internal queues overflow	200
<i>The various types of internal queues and how they might overflow 200 . Using Storm's debug logs to diagnose buffer overflowing</i>	201
8.5 Addressing internal Storm buffers overflowing	203
<i>Adjust the production-to-consumption ratio 203 . Increase the size of the buffer for all topologies 203 . Increase the size of the buffer for a given topology 204 . Max spout pending</i>	205
8.6 Tweaking buffer sizes for performance gain	205
8.7 Summary	206

9 *Trident* **207**

9.1 What is Trident?	208
<i>The different types of Trident operations 210 . Trident streams as a series of batches</i>	211
9.2 Kafka and its role with Trident	212
<i>Breaking down Kafka's design 212 . Kafka's alignment with Trident</i>	215
9.3 Problem definition: Internet radio	216
<i>Defining the data points 217 . Breaking down the problem into a series of steps</i>	217
9.4 Implementing the internet radio design as a Trident topology	217
<i>Implementing the spout with a Trident Kafka spout</i>	219
<i>Deserializing the play log and creating separate streams for each of the fields 220 . Calculating and persisting the counts for artist, title, and tag</i>	224
9.5 Accessing the persisted counts through DRPC	229
<i>Creating a DRPC stream 230 . Applying a DRPC state query to a stream 231 . Making DRPC calls with a DRPC client</i>	232
9.6 Mapping Trident operations to Storm primitives	233

9.7 Scaling a Trident topology	239
<i>Partitions for parallelism 239 • Partitions in Trident streams</i>	240
9.8 Summary	243
<i>afterword</i>	244
<i>index</i>	247