A Case for Understanding End-to-End Performance of Topic Detection and Tracking Based Big Data Applications in the Cloud

Meisong Wang[1], Rajiv Ranjan[2,5(✉)], Prem Prakash Jayaraman[3], Peter Strazdins[1], Pete Burnap[4], Omer Rana[4], and Dimitrios Georgakopulos[3]

[1] Australian National University, Canberra, ACT 2601, Australia
dean.wang@gmail.com
[2] Newcastle University, Newcastle Upon Tyne, UK
rranjans@gmail.com
[3] RMIT University, Melbourne 3000, Australia
[4] Cardiff University, Cardiff, UK
[5] CSIRO, Canberra, Australia

Abstract. Big Data is revolutionizing nearly every aspect of our lives ranging from enterprises to consumers, from science to government. On the other hand, cloud computing recently has emerged as the platform that can provide an effective and economical infrastructure for collection and analysis of big data produced by applications such as topic detection and tracking (TDT). The fundamental challenge is how to cost-effectively orchestrate these big data applications such as TDT over existing cloud computing platforms for accomplishing big data analytic tasks while meeting performance Service Level Agreements (SLAs). In this paper a layered performance model for TDT big data analytic applications that take into account big data characteristics, the data and event flow across myriad cloud software and hardware resources. We present some preliminary results of the proposed systems that show its effectiveness as regards to understanding the complex performance dependencies across multiple layers of TDT applications.

Keywords: Cloud computing · Big data · Hadoop map reduce

1 Introduction

Big Data is revolutionizing nearly every aspect of our lives ranging from enterprises to consumers, from science to government. Managing large, heterogeneous and, rapidly increasing volumes of data has long been a challenge. On the other hand, cloud computing [2,13] in recent times has emerged as the platform that can provide an effective and economical infrastructure for collection and analysis of big data produced by data analytics applications such as topic detection and tracking (TDT). TDT applications detect events (such as disease outbreak, sentiments of customers for certain products or movies etc.) by analysing data

B. Mandler et al. (Eds.) IoT 360° 2015, Part I, LNICST 169, pp. 315–325, 2016.
DOI: 10.1007/978-3-319-47063-4_33

from social media and other online sources. Though big data processing and analytics technologies such as hadoop and mahout have evolved, we still lack orchestration techniques for developing scalable TDT applications in domains such as disease outbreak and sentiment analysis that can elastically scale in response to changing data volume, data velocity, and data variety. Hence, the fundamental challenge is how to cost-effectively orchestrate these TDT applications over cloud-based hardware and software resources for accomplishing big data analytic tasks (e.g. event detection delay) while meeting the new breed of performance Service Level Agreements (SLAs). By the new breed of SLA's we point to the need of future TDT applications that can not be architected to meet the traditional cloud SLA's such as availability and reliability. To the contrary, these new breed of TDT applications need strict SLAs' guarantee on the metrics such as accuracy, precision, and speed of event detection.

To address the above challenges, firstly, it is necessary to establish a taxonomy of performance metrics that can capture the relationship between the applications SLA (e.g., event detection delay, alert generation delay, and alerts sent per second), big data characteristics (e.g. data volume, query rate, and query mix) and resource configuration of the underlying software (e.g., Hadoop, NoSQL, distributed file system, and machine learning library) and hardware (CPU, Storage, and Network). In the literature some performance metric taxonomy and models are available, but they have the following limitations: (i) they target trivial applications (such as "word count") which do not have end-to-end performance management concerns as evident in the complex TDT applications and (ii) most of them are concerned only with the performance modelling the hardware resources while ignoring end-to-end dependencies between the application, software and hardware resource layers. As a consequence, the existing approaches are not appropriate to study the end-to-end performance SLA concerns of the TDT applications. In this vision paper, we propose that a novel, end-to-end taxonomy of performance metrics could be used to develop performance models for studying and analysing the performance SLAs of complex big data applications such as TDT. The novel contributions of this paper include:

- We present a concrete vision statement backed by rigorous analysis of the related work for developing layered and end-to-end performance metric taxonomy for future TDT applications. These performance metrics take into account the data and event flows across multiple software and hardware resource types while considering complex performance dependencies across the layers.
- We present a conceptual architecture for future TDT application which forms the basis for developing the above mentioned performance metrics taxonomy. We also conduct preliminary experimentations for showing the practicality of the proposed approach.

2 Big Data Analytics Application Scenarios

The importance of big data analytics applications such as topic detection [14–17] and tracking has practical values in a variety of fields. Following are some typical application use cases that under take non-real time analysis activities.

1. **Natural Disaster Risk Assessment management application**: By analysing historical data from social media and other sources such as remote sensing satellites and deployed seismic sensors, it is possible to conduct following pre-disaster and post-disaster impact assessment in context of natural disasters such as earthquakes. Historical feeds from social media can be analysed to understand the regions which are most prone to future earthquakes. Such historical feeds can be augmented with crowd sensed data such as high resolution images of public and private infrastructures including buildings, bridges, and roads. These crowd sensed data can be further analysed for pre-assessment of risks and ability of these public and private infrastructures to cope with future earthquakes. The results of such a pre-disaster assessment could be used to evacuate people out of dangerous infrastructures in advance. On the other hand in the post-disaster situations, timely analysis of data from these social media, crowd senses, and other online sources can help rescue teams, medics, and relief workers in planning for future rescue and medical operations.
2. **Traffic pattern analysis application**: By collecting and processing the historical traffic information along with social media feeds, this application can help in meeting the following two goals: Offering the driver the information of the possible traffic congestion; providing advice to the drivers on alternative routes.
3. **Epidemic propagation analysis application**: It is well-known that by carefully analysing the social media feeds related to people's health and well-being could help in learning about past epidemic outbreak. Such data analytic applications can help in improved coordination and deployment of health services.

3 Related Work

The area of performance management of cloud-based big-data processing frameworks have been widely studied. However, understanding and developing an end-to-end performance model of cloud-based big data analytics applications is still in its infancy. In the past several years, Hadoop has been deployed for undertaking batch processing task over large volumes of data (not in real-time). However, most research focus on developing performance model of MapReduce framework only while ignoring the other software and hardware components/resources. In [4] the author describes the complexity of MapReduce (MR) tasks and presents how to model this complexity. Furthermore, the author provides a deep analysis of the working of MapReduce, the interaction and correlation among various steps of MapReduce and the associated costs. The author presents a model to predict the execution time of tasks according to certain cost vectors. The focus of the paper is only on Hadoop more specifically on the MR, HDFS at the IaaS layer. The authors use WordCount, Hive Query Job, and Distributed Pentmino as the usecases for modelling the MapReduce jobs all of whose execution is considerably different from that of analytical machine learning algorithms. *It is well*

understood that machine learning algorithms generate interactive computations and intermediate data that requires a more concrete formulation of the MR execution strategy. In [19], the authors focus on the study of tasks assignment issue in Hadoop. The authors prove that the hadoop task assignment process seeking to minimise the total execution time is NP-Complete. However, this paper is completely based on the mathematical formulation of MR and lacks experimental validations in real world cloud environments.

In [5], the authors propose ARIA (Automated Resource Inference and Allocation) framework composed of SLO (Service Level Object) scheduler, slot estimator, job profiler, profile database and slot allocator to address the challenge of resource allocation for MR jobs to meet the required SLOs requirement bound by a job completion deadline. The applications considered include Word count, Sort, Bayesian classification. TF-IDF, WikiTrends and Twitter. Though ARIA address the challenge of estimating performance of application that use machine learning for analysis, it fails to address the following (1) Provide performance insights across each individual cloud layer i.e. IaaS (CPU, Memory, Network) and PaaS (HDFS storage considerations); (2) Employs a simple online greedy algorithm to calculate the *max*, *min* and *mean* of execution time of Map or Reduce tasks to estimate the MR job execution time and (3) Although they employ bayesian classification, the use of it is limited which renders it insufficient to prove that this performance model will be suitable for cloud-based big data analytics systems. In [18] a task scheduling mechanism for runtime performance management of MR framework is proposed. This management scheduler can use two strategies to allocate resources: the *min-scheduler* and the *max-scheduler*. The min-scheduler will give a job minimum resources to meet certain execution time constraint while the max-scheduler will give high priority jobs maximum resources. The proposed model has been evaluated over applications such as word count and table joins using hive. Similarly, in [6], the authors proposed a framework called *MRShare* which can be plugged into any MapReduce system. The *MRShare* is the first framework to analyse the work sharing problem in MapReduce i.e. different jobs might share resources together. A series of experiments were conducted to validate the proposed work sharing approach on Hadoop frameworks (Hadoop HDFS, MapReduce, Hive and Pig) running on Amazon clusters. In [7], the authors proposed a prediction model for MapReduces performance taking into account the I/O congestion and task failure. The authors design a mathematical model to predict the performance of MapReduce, meanwhile they use the Hadoop as the experimental testbed to validate their model. In [8] the author proposes a prediction model based on greedy policy of MapReduce in terms of different configurations (e.g. different MR parameters, different data sizes, different I/O, etc.).

In [9] the authors propose a performance model of Hadoop which describes data flows and cost information. This performance model can be classified into three parts: Map. Reduce and Shuffle. This performance model involves the CPU cost and the I/O cost. In [10], the authors design a model based on historical job execution records adopting locally weighted linear regression (LWLR) technique

to predict the execution time of a MapReduce job. Similarly, in [12], the authors propose a performance model of Hadoop. This model takes the network into consideration and proves that network bandwidth plays a vital role in the performance of hadoop system. However, these approach does not provide the metrics required to model the end-to-end performance of big data analytics application. For instance, a ML algorithm like the Naive Bayes classification algorithm will have more than one MapReduce task. Furthermore, some steps (such as serialization, etc.) and relations among different steps (such as parallel or overlap) are not considered in the current performance model presented in the literature. Some of related works also study the performance of MapReduce-based machine learning algorithms complexity. However these focus on the theoretical aspects of the algorithm without any evaluation in real cloud environments to validate the theoretical outcomes [11].

3.1 Summary of Limitations

1. The negligence at the Infrastructure-as-a-Service layer: The influence from the memory has been severely neglected. Most performance models of MapReduce or Hadoop (HDFS and MapReduce) are only related to the CPU and job workload, assuming that memory is a trivial aspect. As a matter of fact, it is the other way around. In actuality, the impact of memory on the speed of processing data in particular in big data TDT application needs to be studied in order to develop an effective performance model (e.g. due to the principal of "Spill" operation in the MapReduce process. The Spill means that the output data would be sent to the memory, and only when the memory is near to be filled (no more memory space for storing the new data), the memory starts transmitting data into storage).
2. The negligence from Platform-as-a-Service layer: Consider the example of Hadoop for processing cloud-based TDT applications. The inaccurate assumption of when the shuffle starts could have a significant impact on the system's performance. Many MapReduce model assume that the shuffle part starts when all the Map tasks have been done. However, it is not always the case. In fact, the shuffle process can start before the end of map process by tuning certain performance parameter or metric. Moreover, many models make a simple assumption to determine the number of mapper based only on the size of input data. However, the number of map tasks is mainly controlled by three parameters which are dfs.block.size, mapred.map.tasks and mapred.min.split.size. Hence, determining the number of mapper is a complex task. Finally, at the application layer such as Mahout machine learning component, there is a clear gap in the existent research with respect to performance modelling. We believe, modelling the end-to-end performance of a TDT application involving machine learning components such as Mahout using only MapReduce performance metrics is not accurate. Our preliminary results validate our proposition.
3. Lack of understanding about the dependencies of each software and hardware resources across TDT stack: Considering the Hadoop MapReduce framework,

the relation between components across layer such as memory, the order of processing the data i.e. sequential or MapReduce and the parameters governing the machine learning component (such as Mahout) needs to be analysed and studied to determine the impact they have on each other's performance. This will help in developing a more concrete and accurate end-to-end performance model.

4 End-to-End Performance Modeling of Cloud-Based Big Data Analytics Applications: Our Vision

4.1 Conceptual Architecture

As new TDT applications start to emerge, there is a need for processing high volume, velocity and heterogeneous variety of data (big data characteristics). We need to develop novel application architecture that builds upon the recent progress made in the domain of cloud datacentres offering hardware resources (CPU, Storage, and Network) and big data processing technologies (e.g. Hadoop, Mahout, S4, Spark, NoSQL, etc.) offering software-based application programming abstractions and operations. To this end, we present the conceptual architecture of such a TDT application in Fig. 1. The conceptual architecture consist of three layers including Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). SaaS represents the TDT application, PaaS includes the big data processing technologies or software resources, and IaaS has the cloud datacentre hosted hardware resources.

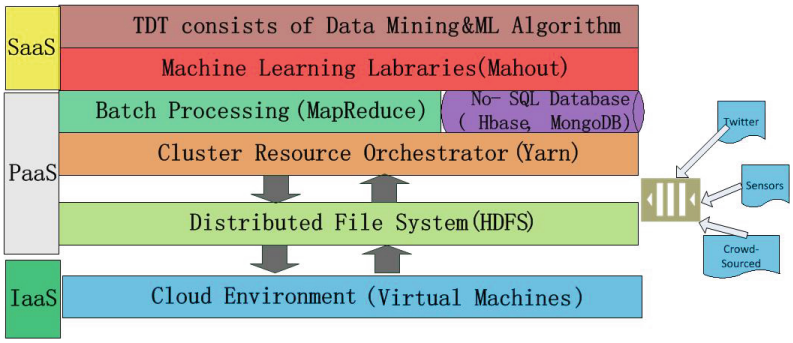

Fig. 1. The conceptual architecture of big data analytics application

4.2 Performance Metric Selection: Preliminary Exploration

Table 1 lists the various high-level parameters that we consider at each layer to build the proposed end-to-end TDT-based big data application performance model. These parameters have been developed taking into consideration the

Table 1. Big data analytics application (TDT) performance metrics for each layer

Layer		Metric
SaaS		Precision and Recall
PaaS	MapReduce	Number of Map tasks, Number of Reduce tasks, the total size of input data, the size of splitting data, the scheduling mechanism, the format of file
	HDFS	The architecture of HDFS like the number of Datanodes, the resources possessed by each node (Namenode and Datanode), the replication number
IaaS		CPU utilization, Memory utilization, Network Bandwidth

conceptual architecture presented in Fig. 1 typical cloud-based batch processing system such as Hadoop.

In the SaaS layer, recall and precision will be impacted considerably due to the different types of data mining algorithms adopted and the variation in data. E.g. considering the naive bayes algorithm, the training part is implemented using four MapReduce steps. However, the testing process consist of only one MapReduce job called *BayesClassifierDriver*. The PaaS layer for the frameworks such as Hadoop has two software components, namely, MapReduce and HDFS. For the MapReduce operation, we will use the number of maps and reducers, the total size of input data, the size of splitting data, the scheduling mechanism (there are three popular kinds of scheduling algorithms which are FIFO - First in First Out, capacity and fair), and file format. For HDFS, we are mainly concerned with the architecture of HDFS. The different architectures of HDFS lead to changes in data storage efficiency that influence the number of replications and storage locations. Another important factor at the HDFS layer is the block size. The block size will directly affect the number of mappers. At the IaaS layer, we consider configurations [20] of CPU, memory and network. The number of CPUs on the shared cluster and the speed of each CPU core has significant impact on the modelling the execution time. Further, memory also plays a pivotal role in determining the execution speed.

5 Preliminary Experimental Outcomes

In this section, we present preliminary experimental trials conducted to validate and verify the correctness of the identified metrics at each layer. The evaluation was conducted using Hadoop 2.4.1 and Mahout 1.0 systems. The performance metric configurations of these system are presented in the Table 2.

In order to validate and verify the identified parameter's influence on various hardware and software components across the layers of a TDT application, we conducted 4 preliminary experiments. Our input dataset is a collection of Tweets related to Flu collected by COSMOS project (www.cs.cf.ac.uk/cosmos/) at Cardiff University. We employ the Naive bayes classification technique to match tweets to topics i.e. *"related to FLU"* and *"not related to flu"*.

Table 2. Experiment test bed configuration

Node	Configuration
Master Node	4 Intel(R) Core(TM) CPU T7700 2.40 GHz 8 GB 10 GB Ubuntu 14
Name Node 1	4 2 Intel(R) Core(TM) CPU T7700 2.40 GHz 4 GB 10 GB Ubuntu 14
Name Node 2	2 Intel(R) Core(TM) CPU T7700 2.40 GHz 4 GB 10 GB Ubuntu 14

In our experiments, we extended the naive bayes implementation provided by the Apache Mahout (mahout.apache.org). Apache Mahout is a scalable machine learning library built on Hadoop Map Reduce framework.

5.1 Experiment 1: Influence of Data Size

In this experiment, we fix the hadoop cluster configuration, while changing the volumes of input Tweet data. We compute and measure the execution time for the following operations (1) converting tweet data into vectors (a requirement for Mahout to process the data) to be consumed by the naive bayes algorithm and (2) training the data for naive bayes classification. The result of this experiment is shown in Table 3.

Table 3. Experiment 1: influence of data size

Data size	Operation	Execution time
0.316 GB	Text vectorisation	4.025 min
	Naive Bayes training	1.444 min
78.34 MB	Text vectorisation	3.98 min
	Naive Bayes training	1.4077 min

5.2 Experiment 2: Changing Hadoop Map Reduce Configuration

Secondly, we conduct experiments with 0.316 GB data keeping the CPU configuration fixed (shown in Table 2) while changing the numbers of mappers and reducers. The result is shown in Table 4 dealing with the execution time of training model.

5.3 Experiment 3: Changing Hadoop Map Reduce Configuration

In this experiment, we use 2 different machine learning algorithms namely the Naive Bayes and C Naive Bayes. The data size used for the experiment is 0.316 GB. The hadoop map reduce layer is configured with 1 mapper and 1 reducer. Table 5 shows the values for Precision and Recall.

Table 4. Experiment 2: changing MR configuration

Mappers	Reducers	Execution time
1	1	1.441 min
2	1	1.219 min
3	3	1.433 min

Table 5. Experiment 3: different machine learning algorithms

Algorithm	Precision	Recall
C-Bayes	59.22 %	61.08 %
Naive-Bayes	57.20 %	60.40 %

Table 6. Experiment 4: changing VM configuration

VM Configuration	Execution time
MapMemory: 1600 Mb; ReduceMemory: 3072 Mb; SortMemory: 512 Mb	1.3328 min
MapMemory: 512 Mb; ReduceMemory: 1024 Mb; SortMemory: 286 Mb	1.4441 min

5.4 Experiment 4: Changing CPU (VM) Configuration

In this experiment, we use 2 different VM configurations for hadoop name and data nodes. The data size used for the experiment is 0.316GB. The hadoop map reduce layer is configured with 1 mapper and 1 reducer and the Mahout layer runs the C-Bayes algorithm. Table 6 shows the outcome of this experiment.

5.5 Experimentation Summary

The experimental outcomes verify the interdependencies between the various components of a big data analytics system across each layer of the cloud and their impact on system's performance. In particular, results of experiment 2 further validates our vision in developing an end-to-end performance model as with more mapper and reduces, the system's performance degraded for an identical dataset and machine learning algorithm.

6 Conclusion

In this paper, we presented our vision and challenges of designing of a generic TDT application (that has non real-time data analytics requirement) based on integrating hardware (CPU, Storage, and Network) and software (batch processing system, NoSQL system, Machine Learning system) resources. We further

considered and analysed the new performance challenges arising in such TDT applications due to integration of multiple resource types and processing of heterogeneous data flows. Next, we developed taxonomy of performance metrics relevant to hardware and resource types. Finally, we conducted small scale experiment based on real-world implementation to study the performance of flu detection TDT application based on varying workload and resource configurations.

In the future work, we will extend our conceptual architecture and performance metric taxonomy to include real-time processing requirements. At the same time we will work on generalizing our taxonomy to include the features of other classes of big data systems such as Apache Spark, and Apache SAMOA (online machine learning library).

References

1. Lara Yejas, O.D., Zhuang, W., Pannu, A.: Big R: large-scale analytics on hadoop using R. In: IEEE International Congress on Big Data (BigData Congress), 27 June-2 July, pp. 570–577 (2014)
2. Yang, X., Sun, J.: An analytical performance model of MapReduce. In: IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 15–17 September, pp. 306–310 (2011)
3. Costa, F., Silva, L., Dahlin, M.: Volunteer cloud computing: MapReduce over the Internet. In: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 16–20 May, pp. 1855–1862 (2011)
4. Lin, X., Meng, Z., Xu, C., Wang, M.: A practical performance model for Hadoop MapReduce. In: IEEE International Conference on Cluster Computing Workshops (CLUSTER WORKSHOPS), pp. 231–239 (2012)
5. Verma, A., Cherkasova, L., Campbell, R.H.: ARIA: automatic resource inference and allocation for mapreduce environments. In: Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC 2011), pp. 235–244. ACM, New York (2011)
6. Nykiel, T., Potamias, M., Mishra, C., Kollios, G., Koudas, N.: MRShare: sharing across multiple queries in MapReduce. Proc. VLDB Endow. **3**(1–2), 494–505 (2010)
7. Cui, X., Lin, X., Hu, C., Zhang, R., Wang, C.: Modeling the Performance of MapReduce under resource contentions and task failures. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2–5 December 2013, vol. 1, pp. 158–163 (2013)
8. Xu, L.: MapReduce framework optimization via performance modeling. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 21–25 May, pp. 2506–2509 (2012)
9. Herodotou, H.: Hadoop performance models. Technical report, CS-2011-05, Computer Science Department, Duke University
10. Khan, M., Jin, Y., Li, M., Xiang, Y., Jiang, C.: Hadoop performance modeling for job estimation and resource provisioning. IEEE Trans. Parallel Distrib. Syst. **27**, 441 (2015)
11. Tamano, H., Nakadai, S., Araki, T.: Optimizing multiple machine learning jobs on MapReduce. In: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 29 November–1 December, pp. 59–66 (2011)

12. Han, J., Ishii, M., Makino, H.: A Hadoop performance model for multi-rack clusters. In: 2013 5th International Conference on Computer Science and Information Technology (CSIT), 27–28 March, pp. 265–274 (2013)
13. Alhamazani, K., Ranjan, R., Mitra, K., Jayaraman, P.P., Huang, Z., Wang, L., Rabhi, F.: CLAMS: cross-layer multi-cloud application monitoring-as-a-service framework. In: IEEE International Conference on Services Computing (SCC), 27 June–2 July, pp. 283–290 (2014)
14. Liu, B., Blasch, E., Chen, Y., Shen, D., Chen, G.: Scalable sentiment classification for big data analysis using Nave Bayes classifier. In: 2013 IEEE International Conference on Big Data, 6–9 October, pp. 99–104 (2013)
15. Amayri, O., Bouguila, N.: Online news topic detection and tracking via localized feature selection. In: The 2013 International Joint Conference on Neural Networks (IJCNN), 4–9 August, pp. 1–8 (2013)
16. Huang, J., Zhao, H., Zhang, J.: Detecting flu transmission by social sensor in China. In: 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on Green Computing and Communications (GreenCom), and IEEE Cyber, Physical and Social Computing, 20–23 August, pp. 1242–1247 (2013)
17. Wu, Z., Liao, J., Zhang, L.: Predicting on retweeting of hot topic tweets in microblog. In: 2013 5th IEEE International Conference on Broadband Network & Multimedia Technology (IC-BNMT), 17–19 November, pp. 119–123 (2013)
18. Berliska, J., Drozdowski, M.: Scheduling divisible MapReduce computations. J. Parallel Distrib. Comput. **71**(3), 450–459 (2011). doi:10.1016/j.jpdc.2010.12.004
19. Fischer, M.J., Su, X., Yin, Y.: Assigning tasks for efficiency in Hadoop: extended abstract. In: Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2010), pp. 30–39. ACM, New York (2010)
20. Zhang, M., Ranjan, R., Nepal, S., Menzel, M., Haller, A.: A declarative recommender system for cloud infrastructure services selection. In: Vanmechelen, K., Altmann, J., Rana, O.F. (eds.) GECON 2012. LNCS, vol. 7714, pp. 102–113. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35194-5_8