PREFACE

The relation of software architectures to functional and quality requirements is of particular importance in cloud computing. Requirements are the basis for deriving software architectures. Furthermore, the environment in which the software will operate is an important aspect to consider in developing high-quality software. That environment has to be taken into account explicitly. One and the same software may be appropriate (e.g., secure) in one environment, but inadequate (e.g., not sufficiently secure) in a different environment. While these considerations are important for every software development task, there are many challenges specific to cloud and big data. In this book, our goal is to collect chapters on systems and architectures for cloud and big data and, more specifically, how software architectures can manage challenges in advanced big data processing.

INTRODUCTION

Software architecture is the earliest design artifact, which realizes the requirements of a software system. It is the manifestation of the earliest design decisions, which comprise the architectural structure (i.e., components and interfaces), the architectural topology (i.e., the architectural style), the architectural infrastructure (e.g., the middleware), the relationship among them, and their relation to other software artifacts (e.g., detailed design and implementation) and the environment. The architecture of a system can also guide the evolution of qualities such as security, reliability, availability, scalability and real-time performance over time. The properties of a particular architecture, whether structural or behavioral, can have global impacts on the software system. Poor architectural realization can threaten the trustworthiness of a system and slow down its evolution. The architectural properties of a system also determine the extent to which it can meet its business and strategic objectives. Consistent with this view is the trend toward focusing software architecture documentation in meeting stakeholder needs and communicating how the software solution addresses their concerns and the business objectives.

Big data is about extracting valuable information from data in order to use it in decision making in business, science, and society. Big data is an emerging paradigm applied to datasets whose size is beyond the ability of commonly used software tools to capture, manage, and process the data within a tolerable elapsed time. Such datasets are often from various sources (Variety), yet unstructured such as social media, sensors, scientific applications, surveillance, video and image archives, Internet texts and documents, Internet search indexing, medical records, business transactions and web logs; and are of large size (Volume) with fast data in/out (Velocity). More importantly, big data has to be of high value (Value) and establish trust in it for business decision making (Veracity). Various technologies are being discussed to support the handling of big data such as massively parallel processing databases, scalable storage systems, cloud computing platforms, and MapReduce. Innovative software architectures play a key role in advanced big data processing.

As the new-generation distributed computing platform, cloud computing environments offer high efficiency and low cost for data-intensive computation in big data applications. Cloud resources and services are available in pay-as-you-go mode, which brings extraordinary flexibility and cost-effectiveness as well as zero investment in the customer's own computing infrastructure. However, these advantages come at a price – people no longer have direct control over their own data. Based on this view, data security becomes a major concern in the adoption of cloud computing.

WHY A NEW BOOK ON SOFTWARE ARCHITECTURE FOR BIG DATA AND THE CLOUD?

We believe that cloud architecture is an emerging and important topic right now. Coupled with the increased number of applications, migrating to mobile devices that make use of cloud storage and cloud software services will see a marked increase in usage, but we are not sure software designers have thought through the changes needed to their applications to use cloud capabilities wisely. Likewise, big data is on everyone's radar right now. Working with big data is tricky, once you get past the knowledge discovery tasks. The real challenge to including big data in data architecture is structuring the data to allow for efficient searching, sorting, and updating (especially if parallel hardware or parallel algorithms are involved).

The area of cloud and big data is rapidly developing, with conferences/workshops exploring opportunities that cloud and big data technology offers to software engineering, both in practice and in research. However, most of these are focused on the challenges imposed by building big data software systems. There is no single resource that brings together research on how software architectures can solve these challenges. The editors of this book have varied and complementary backgrounds in requirements and architecture, specifically in software architectures for cloud and big data. They also have expertise in software engineering for cloud and big data. This book aims to collect together work across different disciplines in software engineering for cloud and big data.

This new book makes a valuable contribution to this existing body of knowledge in terms of state-of-the-art techniques, methodologies, tools, best practices, and guidelines for software quality assurance and points out directions for future software engineering research and practice. This book discusses systematic and disciplined approaches to building software architectures for cloud and big data. We invited chapters on all aspects of software architecture for cloud and big data, including novel and high-quality research related approaches on innovative software development environments and tools for big data processing.

The book provides opportunities for further dissemination of state-of-the-art methods and techniques for representing and evaluating these systems. We asked authors to ensure that all of their chapters will consider the practical application of the topic through case studies, experiments, empirical validation, or systematic comparisons with other approaches already in practice. Topics of interest included, but were not limited to: innovative software architecture for big data processing; theory, frameworks, methodologies, and architecture for cloud and big data; big data technologies; big data visualization and software architectures; innovative software development environments and tools for big data processing; cloud software as a service; software security, privacy with big data; new programming models for big data processing; software simulation and debugging environments for big data processing; research challenges in software architecture for cloud and big data; architecture refactoring for cloud and big data; modeling the software architecture of big data-oriented software systems; and architectures for organizing big data in clouds.

BOOK OUTLINE

In Chapter 1 we present an overview of software architecture for big data and the cloud. The cloud has revolutionized the way we look at software architectures. The emergence of the cloud and its "as-service" layers (e.g., software, platform, databases, infrastructure as services, etc.) has significantly induced the architecture of software systems. Cloud marketplaces, multitenancies, federation, elastic and on-demand access have enabled new modalities to the way we incept, compose, architect, deploy, maintain and evolve architectures of software systems. Properties related to dynamic access of resources; resource pooling; rapid elasticity and utility service provision; economies of scale; dynamicity and multitenancy are arguably the emergent "cloud-architecture significant properties." These properties have influenced not only the behavior of the software systems benefiting from the cloud, but also its structure, style, and topology. It has also moved architecting practices towards architecting for uncertainty, where architecture design decisions are more complex and require us to anticipate the extent to which they can operate in dynamic environments and cope with operational uncertainties and continuous changes. More interestingly, the cloud business model has also moved architecting towards economics-driven architecting, where utilities, risk avoidance, utilization, technical debt monitoring, and optimizing for Service Level Agreements (SLA) are among the business objectives. In this context, architecting in/for the cloud has become an exercise that requires continuous alignments between enterprise and technical objectives. Several architecture styles and architecture-centric development processes that leverage the benefits of the cloud and big data have emerged. The fundamentals of these styles cannot be understood in isolation in what we term as "cloud-architecturally significant requirements." The chapter will review these requirements and explain their implications on architecting for/in the cloud in the presence of big data. It will also roadmap opportunities for researchers and practitioners in software architecture for cloud and big data.

We have divided the rest of the book into five key parts, grouping chapters by their link to these key themes: concepts and models, evaluation of architecture models, big data technologies, resource management, and future directions. Part I papers examine concepts and models. Here the five chapters provide a broad outline of the area of software architectural concepts as applied to big data and the cloud.

PART I: CONCEPTS AND MODELS

Part I of this book consists of five chapters focusing on concepts and models which are useful for understanding big data and cloud computing architectures. Chapter 2, by Ian Groton, discusses issues related to hyperscalability and the changing face of software architecture. Hyperscale systems are pushing the limits of software engineering knowledge on multiple horizons. To address this explosion of data and processing requirements, we need to build systems that can be scaled rapidly with controllable costs and schedules. Hyperscalable systems can grow their capacity and processing capabilities exponentially to serve a potentially global user base, while scaling linearly the resources and costs needed to deliver and operate the system. Successful solutions are not confined to the software architecture and algorithms that comprise an application. Approaches to data architectures and deployment platforms are indelibly intertwined with the software design, and all these dimensions must be considered together in order to meet system scalability requirements. This chapter describes some of the basic principles that underpin system design at scale.

Chapter 3, by Mandy Chessell, Dan Wolfson, and Tim Vincent, discusses different types of systems involved in big data architecture, how the data flows between them, how these data flows intercept with the analytics lifecycle,[1] providing self-service access to data, backed with information governance that creates trust and confidence both to share and consume data. As an industry we need to improve the time to value and success rate of big data projects. This is going to take: better architecture methods that support different big data arenas; tools that automatically manage the metadata and context data necessary to pass data between processing zones; and standard structures for this data to allow for interoperability between cloud services and on premises systems.

Domain-driven design of big data systems based on reference architectures is discussed by Cigdem Avci Salma, Bedir Tekinerdogan, and Ioannis N. Athanasiadis in Chapter 4. Big data has become a very important driver for innovation and growth for various application domains. These application domains impose different requirements on the big data system. Designing a big system as such needs to be carefully considered to realize a system's business goals. In this chapter, the authors have adopted a domain-driven design approach in which they provide a family feature model and reference architecture based on a domain analysis process. The family feature model covers the common and variant features of a broad set of applications, while the reference architecture provides a reusable architecture for deriving concrete application architectures. The authors illustrate their approach by considering Facebook and Twitter as case studies.

Robert Heinrich, Reiner Jung, Christian Zirkelbach, Wilhelm Hasselbring, and Ralf Reussner consider architectural run-time models for quality-aware DevOps in cloud applications in Chapter 5. Cloud-based software applications are designed to change often and rapidly during operations to provide constant quality of service. As a result the boundary between development and operations is becoming increasingly blurred. DevOps is a set of practices for the integrated consideration of developing and operating software. Software architecture is a central artifact in DevOps practices. Architectural information must be available during operations. Existing architectural models used in the development phase differ from those used in the operation phase in terms of abstraction, purpose and content. This chapter presents the iObserve approach to address these differences and allow for phase-spanning usage of architectural models.

In Chapter 6, Tao Chen and Rami Bahsoon present novel ideas for facilitating cloud autoscaling. They examine the similarities between a cloud ecosystem, represented by a collection of cloud-based services with a natural ecosystem. They investigate how the ecological view can be adopted to explain how cloud-based services evolve, and explore the key factors that drive stable and sustainable cloud-based services. To achieve this goal they discuss how to transpose ecological principles, theories and models into autoscaling cloud analogues that spontaneously improve long-term stability and sustainability of a cloud ecosystem.

[1] The data analytics lifecycle is the process that organizes and manages the tasks and activities associated with the analysis of big data.

PART II: ANALYZING AND EVALUATING

The four chapters that make up Part II of this book focus on the analysis and evaluation of several big data and cloud architectural models. The production, processing, and consumption of big data require that all the agents involved in those operations be able to authenticate each other reliably. Authentication of servers and services on the Internet is a surprisingly hard problem. Much research has been done to enhance certificate management in order to create more secure and reliable cloud architectures. However, none of it has been widely adopted, yet. Chapter 7 written by Jiangshan Yu and Mark Ryan provides a survey with critical analysis of the existing proposals for managing public key certificates. Of the three solution categories reviewed, they argue that solutions based on transparent public logs have had the most success in the real world. They present an evaluation framework which should be helpful for future research on designing alternative certificate management systems to secure the Internet.

Performance monitoring in cloud-based big data systems is an important challenge that has not been fully solved, yet. In Chapter 8, Bedir Tekinerdogan, and Alp Oral discuss several potential solutions including caching and scalability. However, none of these approaches solves the problem of disruptive tenants that impede the performance of other tenants. Problems that are difficult to solve using the conventional caching and scalability approaches can be addressed using performance isolation. The authors discuss several performance isolation strategies and describe how the Tork application framework can be used to integrate performance isolation mechanisms found in existing cloud-based big data systems. In this chapter, they propose a framework and a systematic approach for performance isolation in cloud-based big data systems. They present an architectural design for a cloud-based big data system and discuss the integration of feasible performance isolation approaches. They evaluate their approach using PublicFeed, a social media application that is based on a cloud-based big data platform.

Anastasija Efremovska and Patricia Lago discuss the risks and benefits found in software cloud migration in Chapter 9. Multiple factors need to be considered when migrating to the cloud, which include financial, legal, security, organizational, technical risks and benefits, as well as general measures. The body of knowledge on relevant migration factors is mostly neglected in practice and scattered in the scientific literature. Security and legal concerns are no longer considered the greatest issues in cloud migration. Issues like post-migration costs and the potential impact of the migration on organization staff are gaining importance. Migration that was once considered a one-time task has now become a long term project. The authors present a list of factors that can help decision makers in assessing the risks and benefits of moving a software application to the cloud. These can also be used as a base to produce a more complete list of requirements when rearchitecting preexisting software and ease their migration to the cloud.

Big data has only recently been accepted as a recognizable IT discipline. It could be argued that it represents the next major paradigmatic shift in information systems thinking. Big data solutions represent a significant challenge for some organizations. In Chapter 10, Darshan Lopes, Kevin Palmer, and Fiona O'Sullivan present a practitioners perspective on big data. This chapter focuses on four key areas associated with big data that require consideration from a practical and implementation perspective: big data as a new paradigm, product considerations for big data, issues related to using the cloud for hosting big data, and big data implementation frameworks and migration patterns. There is no magic bullet for getting the right big data implementation. But the authors argue that using a combination of business problem focus, open source solutions, power of cloud, and understanding of transition to big data architecture can accelerate the journey and minimize the investment risk.

PART III: TECHNOLOGIES

Technologies used in several big data applications are discussed in the five chapters that make up Part III of this book. In the era of big data, an unprecedented amount of data is generated every second. Real time analytics has become a force for transforming organizations that are looking to increase their consumer base and profit. Real time stream processing systems have gained a lot of attention in social media companies such as Twitter and LinkedIn. In Chapter 11, Xinwei Zhao, Saurabh Garg, Carlos Queiroz, and Rajkumar Buyya propose a taxonomy that can be used to characterize and classify various stream systems. Based on this taxonomy, they compare several open source stream computing platforms. They observe that each platform offers very specific special features that make its architecture unique and that some features make a stream processing platform more applicable than others for different scenarios. The performance of a stream processing system will be always limited by the capacity of the underlying cluster environment where real processing is done. None of the systems surveyed allow the use of cloud computing resources which can scale up and down according to the volume and velocity of data that needs to be processed.

In Chapter 12, Robert Eikermann, Markus Look, Alexander Roth, Bernhard Rumpe, and Andreas Wortmann present a vision of model-driven cloud architecture engineering that relies on reusable service components. This approach enables developers of cloud services and architecture to efficiently build upon existing services. These services exist in the context of ecosystems providing important base services to support reuse, service composition, and user data management. For the latter, the notion of a Digital Me provides benefits for all participating roles: it facilitates data access control and data update for service users and it liberates service developers from providing data management features. Furthermore, it always yields the most up-to-date user data available. Using a generative approach, services for the digitized world can be developed more efficiently on a better suitable, namely higher, level abstraction. This ultimately enables each role, participating in service development, to contribute their domain expertise to dedicated challenges and facilitates service component reuse.

Enterprise applications are data-centric information systems that are being increasingly deployed as Software-as-a-Service (SaaS) Cloud offerings. Such service-oriented enterprise applications allow multiple tenants (i.e., groups of service consumers) to share the computational and storage capabilities of a single cloud application instance. A multitenant SaaS architecture lowers both deployment and maintenance costs. The cost reductions motivate architects to reengineer existing enterprise applications to support multitenancy at the application level. However, in order to preserve data integrity and data confidentiality, the reengineering process must guarantee that different tenants allocated to the same application instance cannot access one another's data. Chapter 13 by Andrei Furda, Colin Fidge, Alistair Barros, and Olaf Zimmermann presents a method and a set of architectural patterns for systematically reengineering data-sensitive enterprise applications into secure multitenant software services that can be deployed to public and private cloud offerings. Architectural refactoring is introduced as a novel reengineering practice and the necessary steps in multitenant refactoring are described from planning to execution to validation (including testing and code reviews). The authors present a realistic case study to illustrate the refactoring process.

Chapter 14 by Stephen Bonner, Ibad Kureshi, John Brennan, and Georgios Theodoropoulos explores the rise of big data and the hardware and software computational strategies that have evolved to deal with this paradigm. Starting with the concept of data-intensive computing, different facets of data processing such as Map/Reduce, machine learning, and streaming data are explored. The evolution of different frameworks such as Hadoop and Spark are outlined, and an assessment of the modular offerings within these frameworks is compared with a detailed analysis of the different functionalities and features. The hardware considerations required to move from compute-intensive to data-intensive are outlined along with the impact of cloud computing on big data. New systems like Apache Spark have been able to push application performance further by utilizing memory locality. The move to in-memory computation has also expanded the number of paradigms these data intensive frameworks are able to perform. The ability to utilize multiple computing paradigms within the same application will result in a new generation of data intensive applications being created.

During recent years, workflows have emerged as an important abstraction for collaborative research and managing complex large-scale distributed data analytics. Workflows are becoming prevalent in distributed environments, such as clusters, grids, and clouds. These environments provide complex infrastructures that aid workflows in scaling and parallel execution of their components. Workflow management systems need to be robust to guard against performance variations and be tolerant against failures. In Chapter 15, Deepak Poola, Mohsen Amini Salehi, Kotagiri Ramamohanarao, and Rajkumar Buyya provide a detailed understanding of faults from a generic viewpoint (e.g., transient, intermittent, and permanent) and a processor viewpoint (such as crash, fail-stop, and byzantine). They also describe several techniques such as replication, provenance, and trust-based approaches used to resolve these faults and provide transparent and seamless experience to workflow users. In addition, they classify various failure models, metrics, tools, and support systems. This chapter provides an insight into failure models and metrics. These can be used by developers to reason about the quality of the schedule and help quantify the fault tolerance of a schedule.

PART IV: RESOURCE MANAGEMENT

Three chapters make up Part IV of this book focusing on managing the architectural resources required for big data and cloud computing. In Chapter 16, Jose Gabriel de Figueiredo Coutinho, Mark Stillwell, Katerina Argyraki, George Ioannidis, Anca Iordache, Christoph Kleineweber, Alexandros Koliousis, John McGlone, Guillaume Pierre, Carmelo Ragusa, Peter Sanders, Thorsten Schütt, Teng Yu, and Alexander Wolf present the HARNESS integrated platform architecture, which is capable of managing resources such as CPUs (Central Processing Units), DFEs (Data Flow Engines), network middle boxes, and SSDs (Solid Straight Drives). The HARNESS architecture is based on a novel hierarchical management approach, designed to make cloud platform systems resilient to new forms of heterogeneity, allowing the introduction of new types of resources without having to redesign the entire system. The HARNESS platform layer automates the process of selecting resources to satisfy application-specific goals (e.g., low completion time and/or low monetary cost) specified by cloud tenants. The HARNESS infrastructure layer is managed by a collection of resource managers, each designed for a specific type of devices incorporated in the HARNESS cloud. The HARNESS enhanced cloud platform stack fully embraces heterogeneity, allowing a more complex and richer context in which to make price/performance tradeoffs and other resource optimizations.

Software development usually relies on a version control system (VCS) to automate the management of source code, documentation, and configuration files. To reduce the cost of storing and managing repositories, data owners often turn to public clouds. Unfortunately, public cloud providers are not necessarily trusted for various reasons. Bo Chen, Reza Curtmola, and Jun Dai introduce the definition of Auditable Version Control Systems (AVCS) in Chapter 17 and instantiate AVCS construction for skip delta-based version control systems which rely on mechanisms to ensure all the versions of a file are retrievable from an untrusted version control server over time. The authors describe an AVCS construction has the following features: (i) the data owner has the ability to check the integrity of all versions in the VCS repository, (ii) the cost of performing integrity checking on all the versions of a file is asymptotically the same as the cost of checking one file version, (iii) it allows the data owner to check the correctness of the version retrieved from the VCS repository, and (iv) it only requires the same amount of storage on the client like a regular (unsecure) VCS system.

Infrastructure-as-a-Service clouds offer access to a scalable virtualized infrastructure on a pay-per-use-basis. This is greatly beneficial for the deployment of scientific workflows. Considerable effort is being made to develop and update existing workflow management systems to support the cloud resource model. The majority of existing systems are designed to work with traditional distributed platforms such as grids and clusters in which the resources are limited and readily-available. In contrast, clouds offer access to elastic and abundant resources that can be provisioned and deprovisioned on-demand. In Chapter 18, Maria A. Rodriguez and Rajkumar Buyya present the use of Workflow Management Systems (WMSs) in cloud computing environments. The authors present a reference architecture for a cloud WMS and explain its key components which include a user interface with workflow modeling tools, submission services, a workflow engine capable of making resource provisioning and scheduling decisions, a set of task and resource monitoring tools, and a set of cloud information services. Efforts to extend an existing workflow system, the Cloudbus WMS, to enable the deployment of scientific applications in cloud computing environments are described. The authors present a case study to demonstrate the added functionality and evaluate the performance and cost of a well-known astronomy application on Microsoft Azure.

PART V: LOOKING AHEAD

In Part V of this book we discuss future directions for software architectural work as applied to big data and cloud computing. The importance of clouds and big data will undoubtedly grow in the future. New technologies will change individual and public life substantially, and enterprises will experience unprecedented new possibilities of operating. As with all new technologies, there are chances and risks, and they have to be applied wisely so that the benefits outweigh the drawbacks. This book reports on the state-of-the-art in software architectures for big data and the cloud. Future developments will build on that state-of-the-art, which constitutes valuable knowledge for all who want to be part of the exciting endeavor of building the digital future.

Ivan Mistrik
Nour Ali
Rami Bahsoon
Maritta Heisel
Bruce R. Maxim