Software Engineering for Big Data Systems

Ian Gorton, Northeastern University
Ayse Basar Bener, Ryerson University
Audris Mockus, University of Tennessee, Knoxville
IT SEEMS DIFFICULT to believe that websites such as YouTube.com (which debuted in November 2005) and Facebook.com (which went public in 2006) have been around for barely a decade. In 2015, YouTube had more than a billion users, who watched 4 billion videos daily and uploaded 300 hours of video per minute.¹ In 2009, Facebook stored 15 billion photos, occupying 1.5 Pbytes and growing by 30 million photos daily. In 2015, Facebook users uploaded 2 billion photos daily, which required 40 Pbytes of new disk capacity daily.² These traffic and storage magnitudes aren’t getting smaller.

The scale of contemporary Internet-based systems, along with their growth rate, is daunting. Data repositories are growing exponentially, and new datacenters hosting tens of thousands of machines are being built around the world to store, process, and analyze this data. Societal change driven by these systems has been immense in the last decade, and the rate of innovation will only grow. We’ve truly entered the era of big data.

**Big Data, Big Challenges**

It’s not only society that’s experiencing profound changes. Big data, together with a host of disruptive technologies including social networking, mobile computing, cloud computing, and the Internet of Things, is speeding up business interactions, shortening project life cycles, and creating immense business opportunities through innovation. In the near future, computational availability will be a commodity, and data and analysis will be provided efficiently and transparently using scalable technologies and tools, and organizations will be driven by the massive, ever-growing datasets they use to solve sophisticated problems.

The exponential growth of data in the last decade has also fueled a new specialization for the software industry: data-intensive, or big data, software systems.³ Internet-born organizations such as Google and Amazon are at this revolution’s cutting edge, collecting, managing, storing, and analyzing some of the largest data repositories ever constructed. Their pioneering efforts, along with those of numerous other big data innovators, have created a variety of open source and commercial data management technologies for any organization to exploit to construct and operate massively scalable, highly available data repositories.

Data-intensive systems have long been built on SQL database technology, which relies primarily on vertical scaling—faster processors and bigger disks—as workload or storage requirements increase. Their inherent vertical-scaling limitations have led to new approaches that relax many core tenets of relational databases. Strictly defined normalized data models, strong data consistency guarantees, and the SQL standard have been replaced by schemaless and intentionally denormalized data models, weak consistency, and proprietary APIs that expose the underlying data management mechanisms to the programmer. These NoSQL⁴ products typically scale horizontally across clusters of low-cost, moderate-performance servers. They achieve high performance, elastic storage capacity, and availability by replicating and partitioning datasets across the cluster. Prominent examples of NoSQL databases include Cassandra, Riak, and MongoDB.

Distributed databases have fundamental quality constraints, defined by Eric Brewer’s CAP theorem.⁵ A system must trade consistency (C—all readers see the same data) against availability (A—every request receives a success or failure response) when a network partition (P—an arbitrary message loss between nodes in the cluster) occurs. The theorem forces software architects and designers to consider complex tradeoffs that carefully balance architectural qualities to achieve application requirements.

Additional design challenges for scalable data-intensive systems stem from five issues. The first is pervasive distribution. Achieving high scalability and availability leads to highly distributed systems. Distribution occurs in all tiers, from webserver farms and caches to backend storage.

The second issue is write-heavy workloads. From social-media sites to high-resolution sensor data collection in the power grid, big data systems must be able to sustain write-heavy workloads. Because writes cost more than reads, you can use data partitioning and distribution (sharding) to spread write operations across...
disks, and you can use replication to provide high availability. However, sharding and replication introduce availability and consistency issues that the system must address.

The third issue is variable request loads. Systems experience highly variable workloads for reasons including product promotions, emergencies, and statutory deadlines such as for tax submissions. To avoid the costs of overprovisioning to handle these occasional spikes, cloud platforms are elastic, letting applications add processing capacity when needed and release resources when the load drops. Effectively exploiting this deployment mechanism requires an architecture that has application-specific strategies to detect an increased load, rapidly add new resources to share the load, and release resources as the load decreases.

The fourth issue is computation-intensive analytics. Most big data systems must support diverse query workloads, mixing requests that require rapid responses with long-running requests that perform complex analytics on significant portions of the data collection. Cost-effectively meeting the different demands of transactional and analytical workloads at large scale is an enormous software engineering design challenge.

The fifth issue is high availability. With thousands of nodes constituting an application, hardware and network failures are inevitable. The resulting distributed software and data architectures must be designed to be resilient. Common approaches for high availability include replicating data across geographical regions, stateless services, and application-specific mechanisms to provide degraded service in the face of failures.

Massive scale also brings a collection of new software engineering challenges. For example, testing becomes hugely problematic in terms of optimizing the test time and resources required. New business requirements might impact hundreds of components, making planning and coordination complex. Engineering teams must be able to work independently to build ever-more sophisticated features; hence centralized management that causes roadblocks simply isn’t effective. And, small optimizations in design and algorithms can lead to major cost reductions.

Addressing these challenges requires careful design tradeoffs that span distributed software, data, and deployment architectures. These tradeoffs also require new software architecture, design, engineering, and deployment approaches. For many software engineers, scalability is becoming a driving requirement for more and more applications. So, we hope this theme issue will be a timely and widely read resource for software engineering professionals and researchers.

In This Issue
This theme issue includes four articles and three essays from industry thought leaders touching several facets of this complicated software engineering puzzle.

Software architecture is a major driver in building big data systems and an area of substantial risk. Big data applications should be able to scale to accommodate increasing data growth while maintaining reliability. In “Strategic Prototyping for Developing Big Data Systems,” Hong-Mei Chen and her colleagues describe and evaluate a risk-aware way to build such systems incrementally. In “A Deep-Intelligence Framework for Online Video Processing,” Weishan Zhang and his colleagues propose an architecture and a framework to handle online-video-processing tasks on a large scale.

Software analytics is necessary to maintain big data systems. This means that data (for example, execution flow data, product usage data, system defect data, and intersystem interactions data) is captured from these applications and analyzed for such purposes as making technical, managerial, and organizational improvements. In “Operational-Log Analysis for Big Data Systems,” Andriy Miransky and his colleagues argue that execution logs and traces are instances of big data that are inevitably generated by big data systems. Analysis of such logs might be the only way to identify an issue with the underlying system. Miransky and his colleagues describe key challenges in log analysis and indicate approaches to address them. In “Building Pipelines for Heterogeneous Execution Environments for Big Data Processing,” Dongyao Wu and his colleagues consider how to integrate multiple heterogeneous op-
erations and analytics tasks. Their pipeline framework lets users perform version control and dependency management in these multiapplication, multilanguage environments.

Finally, the Perspectives department offers essays from three leading innovators in big data systems: Clemens Syzerski from Microsoft, Martin Petitclerc from IBM, and Roger Barga from Amazon. They provide their views on the current and future challenges they face in delivering advanced, Internet-scale services and systems.

We hope you find these articles interesting and stimulating.

References