Modeling and Simulation in Performance Optimization of Big Data Processing Frameworks

Rajiv Ranjan
Commonwealth Scientific and Industrial Research Organization, Australia

Welcome to the fourth installment of "Blue Skies." This department, which will appear six times a year starting with the January/February 2015 issue, will provide in-depth analyses of the most recent and influential research related to cloud computing and big data technologies. In this issue, I'll discuss the role of modeling and simulation science in the era of big data applications. Modeling and simulation can empower practitioners and academics in conducting "what-if" analyses for scheduling policies under variable cloud resource (CPU, storage, and network) configurations, big data processing framework (NoSQL databases, stream processing engines, MapReduce, SQL, and data mining) configurations, and workload (volume, variety, velocity, and query types).

The Big Data Era

According to an IBM study, we're creating 2.5 quintillion (2.5 × 1018) bytes of data every day as of 2012 (www-01.ibm.com/software/data/bigdata). A zettabyte (billion terabytes) of data passed through the Internet in the past year, and the International Data Corporation (IDC) predicts that the digital universe is set to explode to an unimaginable 8 zettabytes by 2015. We're clearly in the era of big data.

Big data is characterized by millions of structured and unstructured datastreams (high velocity), petabytes of historical data (high volume), and heterogeneous data types (high variety). Twitter produces an average of 6,000 tweets per second; however, the number expands to more than 140,000 during certain events (New Year's Eve, the Super Bowl, movie releases, natural disasters, and so on). For example, during the 2010 Haiti earthquake, text messaging via mobile devices and Twitter made headlines for being crucial to emergency responses, but only some 100,000 messages were actually processed by government agencies.[1] In the context of smart energy grids, utility companies are deploying smart meters in homes, offices, and businesses. Moving from the traditional one meter reading a month to automated smart readings once every 15 minutes will lead to 96 million reads per day for every million meters. This could result in a 3,000-fold increase in generated data, which will pose a significant challenge in the real-time diffusion and analysis of data for understanding consumers' energy demand and response patterns.

Such data explosions have led to the next grand challenge in computing: the big data problem,[2-5] which is defined as the practice of collecting complex datasets so large that they're difficult to store, process, and interpret manually or using traditional data management applications (such as Microsoft Excel, relational databases, and data warehousing technologies).

Evolution of Big Data Processing Platforms

The key challenge posed by the big data problem is the ability to process an overwhelming flow of data characterized by the 3Vs—variety, velocity, and volume. Big data sources extend beyond the traditional structured database to include email, sensors, video cameras, social media, and mobile devices (text, video, and audio).

New-generation big data processing technologies include:

- scalable computing infrastructures—such as high-performance and elastic datacenter cloud resources[6]—which provide on-demand access to pay-as-you-go hardware resources (CPU, GPUs, storage, network, and so on);
- data-ingestion frameworks—such as Apache Kafka (http://kafka.apache.org) and Amazon Kinesis (https://aws.amazon.com/kinesis)—which enable high-throughput and low-latency queuing of real-time messages;
- data storage frameworks—such as MongoDB (www.mongodb.org), BigTable,[7] MySQL (www.mysql.com), and Cassandra (http://cassandra.apache.org)—which aid in the management of structured, unstructured, and semistructured data;
- parallel programming frameworks—such as Apache Hadoop (http://hadoop.apache.org) and Apache Storm (http://storm.incubator.apache.org)—which support development of applications for processing historical and streaming data across parallel cluster of cloud resources; and
- scalable data mining frameworks—such as Apache Mahout (http://mahout.apache.org), GraphLab,[8] and MLBase[9]—which implement a wide range of data mining algorithms that can be seamlessly instantiated over parallel programming platforms such as Apache Hadoop.

However, despite the immense potential of existing big data processing platforms, designing, developing, and implementing an optimal big data scheduling platform[10-12] that can guarantee[13-15] performance (minimize response time or latency, maximize throughput) and fault tolerance (maximize availability or reliability) constraints at the same time is challenging, owing to several complexities and uncertainties.

The first complexity is *resource contention and interference*. To minimize infrastructure cost, multiple big data processing frameworks are often hosted on shared cluster computing infrastructures. Sharing cluster resources among heterogeneous big data processing frameworks can also save the huge data migration costs involved in dataflow pipelines. However, such scenarios lead to resource contention and interference as colocated big data processing frameworks will compete for resources and interfere with each other's performance, making it extremely hard to meet performance requirements for real-time decision-making applications such as disaster management, stock purchasing, credit card fraud detection, online patient heart rate monitoring, and traffic management. Although these applications require short response times, current big data application scheduling platforms such as Apache Yarn[10] and Mesos[12] can't guarantee performance because of resource contention, lack of workload prioritization intelligence, and lack of coordinated scheduling capability across multiple big data processing frameworks.

Big data processing frameworks must also deal with *heterogeneous dataflows* (for example, static, streaming, and transactional), heterogeneous data processing semantics (batch processing in Hadoop, continuous stream processing in Storm, and transaction processing in MySQL and Cassandra), and heterogeneous data types (such as unstructured data from Twitter, structured data from traditional SQL databases, and image data from video

The International Data Corporation (IDC) predicts that the digital universe is set to explode to an unimaginable 8 zettabytes by 2015.

cameras) governed by varying data volume, data velocity, and query types. To guarantee performance, scheduling platforms need to be able predict the demands and behaviors of underlying frameworks so they can intelligently distribute and prioritize workloads. Further, it's not clear how such priorities can be preserved across multiple frameworks because dataflows are processed across a distributed platform.

Third, big data processing frameworks must deal with *uncertain resource needs*. Big data processing platforms normally span heterogeneous and distributed software frameworks. These frameworks require heterogeneous and dynamic allocation and configuration of datacenter resources (for example, number and speed of CPUs; storage, cache, and RAM size; and network I/O bandwidth) to accommodate workload changes (3Vs and query types) and to guarantee analytic results within an acceptable delay. Determining an optimal resource configuration for big data processing frameworks is extremely hard because different big data applications have different performance constraints and complexity (3Vs). Current scheduling platforms, such as Apache Yarn and Mesos, entail considerable manual effort, where an administrator has to know in advance how many resources to allocate to each framework without overprovisioning the available resource pool. Further, it's extremely hard to define and aggregate performance constraints of multiple frameworks to get a holistic view of end-to-end performance.

Lack of robustness is another complexity. Big data scheduling platforms such as YARN,[10] Omega,[11] and Mesos[12] can't handle uncertainties arising from failure of datacenter resources, data overloading, malicious attacks, and network link congestion. Most of these scheduling platforms implement a simple failure model, in which a CPU resource instance hosting a big data processing framework (NoSQL or Hadoop, for example) is reconfigured (or restarts, fires a new instance, and so on) and doesn't respond to a certain number of network probes. Such reconfiguration is done without understanding the underlying causes of failures, such as disk failure, processor overload, malicious data, or malicious queries.

Addressing these challenges requires careful consideration of numerous design and performance optimization tasks when developing robust and fault-tolerant big data processing solutions for those applications requiring real-time decision making such as disaster management, stock purchase, credit card fraud detection, and traffic management.

A hard challenge for big data is balancing performance and cost tradeoffs by optimizing configuration at both the hardware and software layers.

How Modeling and Simulation Can Help

A hard challenge for big data is balancing performance and cost tradeoffs by optimizing configuration at both the hardware and software layers to accommodate users' constraints (for example, analytics result delay and alert generation delay) while addressing the four complexities noted. Hence, we need an approach that can help engineers and researchers analyze the impact of these complexities as well as software and hardware configuration interdependencies upon the final performance requirements achievable from a big data application. Conducting such a study in a real computing environment can be challenging for several reasons:

- Procuring or renting a large-scale datacenter resource pool that will accurately reflect realistic application deployment and let practitioners experiment with dynamic hardware and software resource configurations and 3Vs isn't cost effective.
- Frequently changing experiment configurations in a large-scale real testbed involves a lot of manual configuration, making the performance analysis itself time consuming. As a result, reproducing results becomes extremely difficult, making most of the experiments non-repeatable.
- Incorporating and controlling different types of failure behaviors and benchmarks across heterogeneous software and hardware resource types in a real testbed (such as Amazon EC2, Open Cirrus, and Microsoft Azure) environment is extremely hard.

Simulation-based approaches to performance testing and benchmarking offer significant advantages. For example, multiple big data application developers and researchers can perform tests in a controllable and repeatable manner. In addition, it's easier to find performance bottlenecks in a simulated environment than in a real-world test-bed. Simulation-based approaches also simplify experimenting with various hardware resource and big data processing framework configurations and collecting insights about the impact of each design choice on the performance guarantees (service-level agreements). They also let developers and researchers share their simulation datasets and environment setups, leading to better validation of hypothesis and reproducibility of results. Finally, using these approaches, developers and researchers can instantiate multiple big data processing frameworks and diverse workload scenarios.

Distributed System Simulators

Over the last decades, many simulation frameworks have been developed for studying the behavior of large-scale distributed systems for hosting application services (for example, social networking, Web hosting, scientific applications, and content delivery). Popular simulators can be classified based on the distributed system model they are capable of simulating and modeling:

- GridSim,[16] MicroGrid,[17] GangSim,[18] SimGrid,[19] and OptorSim[20] simulate grid computing system models and scheduling algorithms.
- PlanetSim simulates peer-to-peer network models such as structured and unstructured overlay networks.[21] In one study, researchers integrated PlanetSim with GridSim to evaluate the performance of decentralized and coordinated scheduling of scientific applications across multiple computational sites (clusters, supercomputers, and so on).[22]
- CDNSim, developed by extending the OMNet++ library, simulates content delivery networks for studying content management policies (caching, redirection, replica placement, and load balancing).[23]

Although these simulators were widely adopted, they unfortunately don't support modeling and simulation of diverse big data processing frameworks and virtualized datacenter-based cloud resources. Following the tradition of the grid computing era, researchers developed several simulators to facilitate research on various aspects of cloud computing infrastructures.

GreenCloud, a packet-level simulator developed by extending the Network Simulator (NS-2; http://nsnam.isi.edu/nsnam/index.php/User_Information), models behaviors of network links, switches, gateways, and other hardware resources (CPU and storage) in a cloud datacenter.[24] GreenCloud aims to simplify performance tests of energy-aware scheduling algorithms in cloud environments. Because it's a packet-level simulator, it requires extra memory and processing power to create and transmit packets across simulation entities.

MDCSim supports simulation of specified hardware resources in a datacenter from multiple vendors and allows energy consumption profiling for studying scheduling and resource management policies.[25] The current MDCSim release support models for simulating multitier applications consisting of web-server, application server, and database service. It also implements two types of cluster network routing models: InfiniBand architecture and 10-Gigabit Ethernet.

DCSim supports simulation and modeling of CPU resources, data replication policies, and CPU migration policies.[26] However, it doesn't support network topology in lieu of improving scalability. To support dynamic application component migration studies, DCSim implements features for virtual machine (VM) live migration and replication. Because the application entity is only implemented at an abstract level, users must implement specific cloud application models.

CloudSim is one of the most widely used discrete-event simulation frameworks because it's highly extensible and flexible.[27] It provides models for all hardware resources including CPUs (VMs), storage, and networks (network contention and delays) within multiple datacenters. The network simulation in CloudSim is built upon the BRITE network topology generator and communication model.[28] CloudSim has extensive support for application scheduling level simulation (such as for scientific and Web hosting), because it provides cloud broker and cloud exchange (for federated datacenter resource pooling) entities.

MR-CloudSim is an extension of CloudSim for simulating MapReduce big data processing models.[29] However, MR-CloudSim only supports simplistic, single-state map and reduce computations. Further, it lacks support for network link modeling, which is a critical element affecting the performance of MapReduce applications.

To support provider-specific analysis of application performance, CDOSim[30] extends CloudSim and integrates with the cloud migration framework (CloudMIG).[31] Unfortunately, CDOSim and CloudMIG are based on the enterprise resource planning system model, whose computational, data storage, data processing, and software modeling needs fundamentally differ from those of big data applications and frameworks.

Big Data Framework Benchmarks

Fueled by the need to analyze the performance of different big data processing frameworks, researchers have introduced several benchmarks, including BigDataBench,[32] BigBench,[33] Hibench,[34] PigMix, CloudSuite,[35] and GridMix. These benchmark suites model workloads for stress testing one or more categories of big data processing frameworks. For example, Hibench has Sort, WordCount, TeraSort, PageRank, K-means, and Bayes classification workloads for loading Hadoop and Hive frameworks. Among these frameworks, BigDataBench is most comprehensive as it constitutes workload models for NoSQL, DBMS, SPEs, and batch processing frameworks. Primarily, BigDataBench targets the search engine, social network, and e-commerce application domains.

Despite the significant progress, we still need a holistic, comprehensive simulation platform that lets us analyze big data application scheduling across heterogeneous big data processing frameworks and hardware resources. Hence, future research efforts need to focus on developing an integrated simulation and benchmarking framework that can support modeling of both heterogeneous data programming abstractions such as MapReduce in Hadoop, continuous query operators in Storm, and transactional operators in MySQL and Cassandra; and heterogeneous dataflows (for example, static, streams, and transactions), workload processing (batch processing in Hadoop, continuous stream processing in Storm, and transaction processing in MySQL and Cassandra), and hardware resource configurations. It must also support evaluation templates that incorporate details on application-level performance constraints, fault-injection models, big data processing benchmarks and configurations relevant to specific application types (such as credit card fraud detection and emergency management). Finally, such a framework must support failure injection models at both the software and hardware layer.

References

1. J. Heinzelman and K. Baptista, "Effective Disaster Response Needs Integrated Messaging," SciDevNet, 16 Nov. 2012; www.scidev.net/en/new-technologies/icts/opinions/effective-disaster-response-needs-integrated-messaging.html.
2. X. Wu et al., "Data Mining with Big Data," *IEEE Trans. Knowledge and Data Eng.*, vol. 26, no. 1, 2014, pp. 97–107.
3. Z. Deng et al., "Parallel Processing of Dynamic Continuous Queries over Streaming Data Flows," *IEEE Trans. Parallel and Distributed Systems*, preprint, doi:10.1109/TPDS.2014.2311811.
4. W. Fan and A. Bifet, "Mining Big Data: Current Status, and Forecast to the Future," *SIGKDD Explorations Newsletter*, vol. 14, no. 2, 2013, pp. 1–5.
5. R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," *IEEE Cloud Computing*, vol.1, no.1, 2014, pp. 78–83.
6. L. Wang et al., eds., *Cloud Computing: Methodology, System, and Applications*, CRC Press, Taylor & Francis, 2011.
7. F. Chang et al., "Bigtable: A Distributed Storage System for Structured Data," *ACM Trans. Computer Systems*, vol. 26, no. 2, 2008, article 4.
8. Y. Low et al., "Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud," *Proc. VLDB Endowment*, vol. 5, no. 8, 2012, pp. 716–727.
9. T. Kraska et al., "MLbase: A Distributed Machine-Learning System," *Proc. 6th Biennial Conf. Innovative Data Systems Research* (CIDR 13), 2013.
10. V. Kumar et al., "Apache Hadoop YARN: Yet Another Resource Negotiator," *Proc. 4th Ann. Symp. Cloud Computing* (SOCC 13), 2013, article 5.
11. M. Schwarzkopf et al., "Omega: Flexible, Scalable Schedulers for Large Compute Clusters," *Proc. 8th ACM European Conf. Computer Systems* (EuroSys 13), 2013, pp. 351–364.
12 B. Hindman, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," *Proc. 8th USENIX Conf. Networked Systems Design and Implementation* (NSDI 11), 2011, pp. 295–308.
13. R. Zhang et al., "Getting Your Big Data Priorities Straight: A Demonstration of Priority-Based QoS Using Social-Network-Driven Stock Recommendation," *Proc. 40th Int'l Conf. Very Large Data Bases* (VLDB 14), 2014, pp. 1665–1668.
14. M. Kunjir et al., "Thoth: Towards Managing a Multi-System Cluster," *Proc. 40th Int'l Conf. Very Large Data Bases* (VLDB 14), 2014.
15. A. Simitsis et al., "Optimizing Analytic Data Flows for Multiple Execution Engines," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 12), 2012, pp. 829–840.
16. R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," *J. Concurrency and Computation: Practice and Experience*, vol. 14, nos. 13–15, 2002, pp. 1175–1220.
17. H.J. Song et al., "The MicroGrid: A Scientific Tool for Modeling Com-

putational Grids," *Proc. ACM/IEEE Conf. Supercomputing* (SC 00), 2000, article 53.
18. C.L. Dumitrescu and I. Foster, "GangSim: A Simulator for Grid Scheduling Studies," *Proc. 5th IEEE Int'l Symp. Cluster Computing and the Grid* (CCGrid 05), vol. 2, 2005, pp. 1151–1158.
19. H. Casanova, "Simgrid: A Toolkit for the Simulation of Application Scheduling," *Proc. 1st Int'l Symp. Cluster Computing and the Grid* (CCGRID 01), 2001, pp. 430–437.
20. W.H. Bell et al., "Simulation of Dynamic Grid Replication Strategies in OptorSim," *Proc. 3rd Int'l Workshop Grid Computing* (GRID 02), 2002, pp. 46–57.
21. P. García et al., "PlanetSim: A New Overlay Network Simulation Framework," *Proc. 4th Int'l Conf. Software Eng. and Middleware* (SEM 04), 2004, pp. 123–136.
22. R. Ranjan, A. Harwood, and R. Buyya, "Coordinated Load Management in Peer-to-Peer Coupled Federated Grid Systems," *J. Supercomputing*, vol. 61, no. 2, 2012, pp. 292–316.
23. K. Stamos et al., "CDNsim: A Simulation Tool for Content Distribution Networks," *ACM Trans. Modelling and Computer Simulation*, vol. 20, no. 2, 2010, article 10.
24. D. Kliazovich et al., "GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers," *Proc. Global Telecomm. Conf.* (GLOBECOM 10), 2010, pp. 1–5.
25. S.H. Lim et al., "MDCSim: A Multi-Tier Data Center Simulation, Platform," *Proc. IEEE Int'l Conf. Cluster Computing and Workshops*, 2009, pp. 1–9.
26. C. Chen, Y. Liu, and R. Chang, "DC-Sim: Design Analysis on Virtualization Data Center," *Proc. 9th Int'l Conf. Ubiquitous Intelligence and Computing and 9th Int'l Conf. Autonomic and Trusted Computing* (UIC-ATC 12), 2012, pp. 900–905.
27. R.N. Calheiros et al., "CloudSim: A Toolkit for the Modeling and Simulation of Cloud Resource Management and Application Provisioning Techniques," *J. Software: Practice and Experience*, vol. 41, no. 1, 2011, pp. 23–50.
28. A. Medina et al., "BRITE: An Approach to Universal Topology Generation," *Proc. Int'l Workshop on Modeling, Analysis and Simulation of Computer and Telecomm. Systems* (MASCOTS 01), 2001, pp. 346–353.
29. J. Jung and H Kim, "MR-CloudSim: Designing and Implementing MapReduce Computing Model on CloudSim," *Proc. Int'l Conf. ICT Convergence* (ICTC 12), 2012, pp. 504–509.
30. F. Fittkau, S. Frey, and W. Hasselbring, "CDOSim: Simulating Cloud Deployment Options for Software Migration Support," *Proc. IEEE 6th Int'l Workshop Maintenance and Evolution of Service-Oriented and Cloud-Based Systems* (MESOCA 12), 2012, pp. 37–46.
31. S. Frey, W. Hasselbring, and B. Schnoor, "Automatic Conformance Checking for Migrating Software Systems to Cloud Infrastructures and Platforms," *J. Software Maintenance and Evolution: Research and Practice*, vol. 25, no. 10, 2012, pp. 1089–1115.
32. L. Wang et al., "BigDataBench: A Big Data Benchmark Suite from Internet Services," *Proc. 20th IEEE Int'l Symp. High-Performance Computer Architecture* (HPCA 14), 2014, pp. 488-499.
33. A. Ghazal, "BigBench: Towards an Industry Standard Benchmark for Big Data Analytics," *Proc. 2013 ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 13), 2013, pp. 1197–1208.
34. S. Huang et al., "The HiBench Benchmark Suite: Characterization of the MapReduce-based Data Analysis," *Proc. IEEE 26th Int'l Conf. Data Eng. Workshops* (ICDEW 10), 2010, pp. 41–51.
35. M. Ferdman et al., "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware," *SIGARCH Computer Architecture News*, vol. 40, no. 1, 2012, pp. 37–48.

RAJIV RANJAN *is a senior research scientist, Julius Fellow, and project leader at the Commonwealth Scientific and Industrial Research Organization. At CSIRO, he leads research projects related to cloud computing, content delivery networks, and big data analytics for Internet of Things (IoT) and multimedia applications. Ranjan has a PhD in computer science and software engineering from the University of Melbourne.*