Blind Men and an Elephant Coalescing Open-source, Academic, and Industrial Perspectives on BigData

Chris Douglas [1], Carlo Curino [2]

Microsoft
Cloud and Information Services Lab
[1] cdoug@microsoft.com, [2] ccurino@microsoft.com

***Abstract*—This tutorial is organized in two parts. In the first half, we will present an overview of applications and services in the BigData ecosystem. We will use known distributed database and systems literature as landmarks to orient the attendees in this fast-evolving space. Throughout, we will contrast models of resource management, performance, and the constraints that shape the architectures of prominent systems. We will also discuss the role of academia and industry in the development of open-source infrastructure, with an emphasis on open problems and strategies for collaboration. We assume only basic familiarity with distributed systems.**

In the second half, we will delve into Apache Hadoop YARN. YARN (Yet Another Resource Negotiator) transformed Hadoop from a MapReduce engine to a general-purpose cluster scheduler. Since its introduction, it has been deployed in production and extended to support use cases beyond large-scale batch processing. The tutorial will present the active research and development supporting such heterogeneous workloads, with particular attention to multi-tenant scheduling. Topics include security, resource isolation, protocols, and preemption. This portion will be detailed, but accessible to anyone with a background in distributed systems and all attendees of the first half of the tutorial.

I. Introduction

Early web companies were the first to face the scalability limits of existing data management solutions, when trying to apply them to the indexing of the web. To tackle this new challenge, they build ad-hoc data technology to store [1], [2] and process [3], [4], [5], [6] massive amounts of data.

Since this early efforts, insights extracted from petabytes of data and metadata have driven decisions not only at the web companies that spawned this early BigData technologies, but also in industries such as finance, medicine, manufacturing, and retail. While the scale of the web called for novel implementations, one can recognize techniques from the databases and operating systems these systems are modeled on—this is one of the implicit goals of our tutorial.

The explosive success of BigData technologies, led to a quick growth of a broad range of use cases, often well beyond the initial design of system like Hadoop. These authors' favorite "abuse" of early BigData infrastructures is the spawning of a webserver farm as a map-only job [7] atop Hadoop mapreduce. This testify the need for scalable cluster management functionalities, in this sense MapReduce has been often used as a quick way to access large cluster resources, by completely bypassing/hijacking the programming model.

The implicit demand for a "cluster OS" that safeguards and manages data, allocates resources among tenants, recovers from faults, and scales horizontally on commodity hardware has driven development of many competing, layered solutions [8], [9], [10], [11]. The Apache Hadoop project [4] has been among the technologies at the center of this revolution. From its origins as a distributed filesystem (HDFS) and MapReduce implementation building Yahoo's search index, it has been the keystone species in the open-source BigData ecosystem. As that ecosystem matured, the core scheduling and resource management logic in MapReduce was generalized and released as YARN [8]. In contrast to web-scale analytics engines that prescribe a single interface like MapReduce, YARN supports a heterogeneous mix of frameworks and enforces a set of operator invariants on allocation. Our tutorial will describe the current state of the framework, its pedigree, and put its future development in context.

YARN is far from the only solution proposed in this space, and there is no consensus that its decomposition of the problem is optimal. Many schedulers for HPC workloads have been in development for decades [12], [13], [14], [15], targeting a diverse set of scientific and enterprise deployments. The Apache Mesos project [11] also targets BigData analytic workloads, though in contrast to YARN, it considers frameworks as the unit of concern to cluster operators, rather than tenants running applications. Other researchers proposed evolutions of the Mesos model [10], that lifted some of its early limitations.

Parallel databases have not placidly accepted the premise of BigData analytics engines [16], developing a range of techniques and extensions to reduce the gap between the two models. Some approaches [17] leveraged traditional DBMS as a single-node building block, and leveraged Hadoop technologies to handle fault-tolerance and distributions, other built novel stacks that are more closely inspired by the MPP database design, while providing strong integration in Hadoop ecosystem [18], [19], [20]. Moreover, all the DB vendors hurried to provide tooling, and infrastructures to ease ingestion, and exporting to and from the Hadoop ecosystem. Our tutorial will attempt to create a taxonomy of these approaches,

978-1-4799-7964-6/15/$31.00 © 2015 IEEE

contrasting the tradeoffs and assumptions that distinguish each offering.

A. Blind Men and the Elephant

The speakers in this tutorial approach the world of BigData from different backgrounds (academia vs industry, DB vs systems). In this tutorial, we work through our biases (engaging the audience in our disagreements) and explore this space from both angles. This is a core theme of the tutorial, hence the ironic title "Blind Men and an Elephant", [1], from an old, Indian story of a group of blind men touching an elephant and comparing notes to understand its nature.

We follow Apache Hadoop evolution as a landmark for discussing the contributions of industry, academia, and open-source communities to the development of BigData.

As an *industry* byword, Hadoop is indispensable infrastructure. The lessons learned in building databases and operating systems are applied to modern Hadoop development. By way of example, the explosion of SQL engines have brought staples of database development such as snapshot isolation [21], modern columnar formats [22], [23], and cost-based query optimization [24] to BigData clusters. The workflow managers sufficient for large-scale analytic workloads are being supplemented with the security [25], [26], [27] and manageability [28] expected by enterprises.

As an *open-source* project, Hadoop has several unique challenges. Many projects' agendas are set by groups with conflicting interests, often as they compete with one another in the market. On the other hand, the community fosters very high speed development, with many companies contributing to a single ecosystem, and enforces code quality and software engineering principles. Excluding rare cases, the core technical value of a solution is the strongest currency and main motivator for the community—this is akin to the intellectual honesty that characterizes most scientific endeavors, and thus an ideal target of academic contributions.

As a target and consumer of *research*, the BigData ecosystem around Hadoop has relied on some technology [29], rebuilt others [30], and accepted contributions [31], [32] directly. Due to its open-source code, its abundant reference and training literature, and specialized execution engine, academics have found myriad ways to improve, tweak, and reappropriate the system to novel ends.

All of these constituencies shape the technology that Hadoop is becoming. We will provide our perspective on how each has contributed to the project, and where we see that influence directing the course of this ecosystem in the future.

II. Part One

A. Foundations

Yahoo! needed to replace the infrastructure driving its WebMap application, the technology in search that builds a graph for the known web. In 2006, it contained more than 100 billion nodes and 1 trillion edges. The old infrastructure, named "Dreadnought," [33] had reached the limits of its scalability at 800 machines and a significant shift in its architecture was required to match the pace of the web. Dreadnought already executed distributed applications that resembled MapReduce programs, so by implementing or adopting a more scalable MapReduce framework, significant parts of the search pipeline could be migrated easily.

Though it had started development of an internal implementation, Yahoo! elected to abandon that effort in favor of an existing, open-source implementation. Doing so offered several strategic advantages against its competitors in search. Most notably, Yahoo! hoped to create a platform that would commoditize the costly analytics infrastructure that its competitors maintained themselves. In this, adopting and driving Apache Hadoop through its infancy achieved this aim beyond any reasonable expectation.

However, software written for the specialized environment of industrial web search pipelines applies unnaturally to the modern, heterogeneous "data lake." Support for ad hoc queries, streaming, serving, graph analytics, and other applications was not part of the MapReduce paradigm. The gravity of the data stored in the cluster offered few alternatives to users, who had to translate their queries to the cluster API. Academic efforts [34], [35], [36], [37], [38], [39], [40] and high-level languages built on dataflow engines [6], [5], [41], [42] struggled to extract performance from iterative machine-learning applications, to express resource constraints, to optimize for known properties of the underlying data, and to experiment with new features and models.

Separating the programming model from the resource allocator is a natural design, with many precedents in HPC. In fact, early deployments of Hadoop would use HPC schedulers (Torque and Maui) to create ad hoc MapReduce clusters on top of a persistent HDFS instance through a wrapper interface called HoD (Hadoop on Demand). At Berkeley, software that would become Apache Mesos [11] supported not only ad hoc Hadoop clusters, but heterogeneous frameworks. Mesos traces its pedigree to infrastructure deployed at Google, and its development tracks Omega [10], an evolution of that architecture. Concurrent with development of YARN, Facebook built Corona [9], a federated MapReduce deployment targeting low latency queries.

The deployment environment also changed significantly since the early days of the software. When Hadoop was designed, nodes comprising a typical cluster of inexpensive, commodity hardware had far fewer spindles (iops), memory, and CPU cores than a modern machine. Datacenters with 10GigE bisection bandwidth are increasingly common, unlike the 10:1 oversubscribed fattree networks of earlier datacenters, causing some to question the relevance of optimizing for locality, once the cornerstone of MapReduce scheduling.[43] The declining cost of SSDs and the prospect of non-volatile RAM has been a rich source of speculation and prototypes.

[1] See http://en.wikipedia.org/wiki/Blind_men_and_an_elephant

B. Ecosystem

As MapReduce became a library running in a YARN cluster, the diversity of applications running in the same cluster exploded. Almost immediately, several meta-frameworks have emerged to support developers on the platform. Among them, Apache REEF[44] and Apache Twill [45] are incubating projects at the Apache Software Foundation.[2] Both projects offer a layer of abstraction over the resource allocation layer. Apache Tez [46] has all but replaced MapReduce as the default execution substrate in Hadoop clusters. Frameworks formerly built on MapReduce— notably Pig [6], Hive [5], and Cascading [47]— offer a Tez engine. Tez is modeled on the Dryad [30] engine that executes arbitrary DAGs.

While BigData systems distinguished themselves by processing unstructured data, many data warehouses contain repositories of curated, structured data. The list of open-source [19], [48], [5], [49], [50], [51], [52], [53], [20], [54] and proprietary [55], [56] SQL engines have brought decades of database research to the BigData ecosystem. In this tutorial, we will cover some of the techniques that have migrated to process structured data, such as columnar techniques and formats such as ORC[22] and Parquet [23].

Cluster pipelines also require tools for data management, particularly for coordinating pipelines, managing access control, and enforcing policies. Apache Oozie [57], Falcon [28], Sentry [26], Knox [25], and Ranger [27] are all examples of the passive "data management" stack that is essential to cluster operations.

We will also briefly cover some of the services deployed alongside managed clusters, such as HBase, Cassandra, and deployment shims such as Slider [58].

III. Part Two

A. YARN

YARN is composed of the following components. A *resource manager* (RM) that assigns resources to applications, according to the invariants set by the cluster operator. The most prominent schedulers are the fair scheduler [59] and the capacity scheduler [60], both of which maintain weighted guarantees across cluster tenants.

The RM receives requests for resources from a special coordinator process for each job or service, called the *application master* (AM). Most often, an AM is created by the RM in the cluster, and it is responsible for requesting cluster resources to do work. By way of example, a MapReduce AM will load a description of the job and attempt to schedule each *map* and *reduce* task close to its data. It does this by registering with the RM, building a model of its pending work, and submitting a *resource request* describing the hard and soft constraints for resources in the cluster. The RM will respond with a lease bound to a particular node, granting the AM permission to start a *container* that occupies those resources.

[2]"Incubation" is how projects learn how to operate according to that foundation's charter as provisionally accepted communities.

The AM starts the process by describing the other resources (binaries, local libraries, etc.) and binding this description to the lease. It then submits the description to a *node manager* (NM), a daemon running on every worker in the cluster. The NM will validate the authenticity of the lease, download the container dependencies, start the process, and provide services to the container.

In this tutorial, we will describe the container lifecycle in detail, with particular attention to some of the open problems and active development in YARN. We will also contrast the design choices in YARNwith other frameworks, to highlight the tradeoffs made by designers and application authors.

As YARN evolves to cover long-running services [58], resource-aware scheduling [61], preemption/suspend of applications [8], capacity reservation [32], and distributed scheduling [62], [63], [64], it covers territory familiar to academia and industry. In this tutorial, we will discuss how these features are taking shape and offer our opinion on where the platform is headed.

B. Research

We equate the world of BigData to the early DBMS products of the 1970s: widely adopted and far from mature, where 100X performance improvements are still common. Particularly because it is open-source software, academia has published myriad "better than Hadoop" papers documenting substantial gains against the MapReduce implementation. Many of these papers criticize domains for which MapReduce is a poor fit, proposing simplistic solutions that lack practical appeal. Others have dug much deeper, and exposed fundamental limitations in the existing BigData landscape.

As part of its mission, the Apache community is open to contributions and novel approaches. When a feature imposes a substantial change to core systems (e.g., preemption features in YARN), an implementation sensitive to its context will often be accepted. For its openness, we consider Hadoop one of the best vehicles to amplify the real-world impact of academic research. In the tutorial, we will present some of our own research [8], [32], [44] in resource management, mostly as a running example of how it is possible to engage with the community.

Finally, a large amount of research has focused on supporting Machine Learning workloads [65], [66], [67], [68], [69]. The key characteristics of these computations lead to the emergence of a handful of key optimizations common across several systems. As an example, several systems trade the MapReduce fault-tolerance model for more deliberate materialization of intermediate results (e.g., pipelining), and will reuse processes and memory-resident datasets to accelerate computation. In the tutorial, we will focus on this systems aspect and highlight the key, common features and strengths of these classes of systems.

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *SOSP*, 2003.

[2] D. Borthakur, "Hdfs architecture guide," *HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf*, 2008.

[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, 2008.

[4] A. C. Murthy, C. Douglas, M. Konar, O. O'Malley, S. Radia, S. Agarwal, and V. KV, "Architecture of next generation apache hadoop mapreduce framework," Tech. rep., Apache Hadoop, Tech. Rep., 2011.

[5] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive – a warehousing solution over a map-reduce framework," in *PVLDB*, 2009.

[6] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *SIGMOD*, 2008.

[7] S. Krishnan and J. C. Counio, "Pepper: An elastic web server farm for cloud based on hadoop," in *CloudCom*, 2010.

[8] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache hadoop yarn: Yet another resource negotiator," in *SOCC*, 2013.

[9] Facebook Engineering. (2012) Under the Hood: Scheduling MapReduce jobs more efficiently with Corona.

[10] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: flexible, scalable schedulers for large compute clusters," in *ACM European Conference on Computer Systems*, 2013.

[11] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *NSDI*, 2011.

[12] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, 2003.

[13] Tannenbaum and et. al., "Condor: a distributed job scheduler," in *Beowulf cluster computing with Linux*, 2001.

[14] "Maui Scheduler Open Cluster Software," http://mauischeduler. sourceforge.net/.

[15] G. Staples, "Torque resource manager," in *IEEE SC*, 2006.

[16] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "Mapreduce and parallel dbmss: Friends or foes?" *Commun. ACM*, 2010.

[17] A. Abouzeid and et al., "Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads," *PVLDB*, 2009.

[18] S. Altwaijry and et al., "Asterixdb: A scalable, open source bdms," *PVLDB*, 2014.

[19] M. Kornacker and et al., "Impala: A modern, open-source sql engine for hadoop," 2015.

[20] "Presto: Distributed sql query engine for big data," Presto developer community, 2014, http://prestodb.io.

[21] "Implement insert, update, and delete in Hive with full ACID support," https://issues.apache.org/jira/browse/HIVE-5317.

[22] "Create a new Optimized Row Columnar file format for Hive," https://issues.apache.org/jira/browse/HIVE-3874.

[23] S. Melnik and et al., "Dremel: Interactive analysis of web-scale datasets," *PVLDB*, 2010.

[24] Calcite developer community (formerly Optiq), "Calcite: Apache incubator project," 2014, https://github.com/apache/incubator-calcite.

[25] "Apache Knox," https://knox.apache.org.

[26] "Apache Sentry," https://sentry.incubator.apache.org.

[27] "Apache Ranger," https://ranger.incubator.apache.org.

[28] "Apache Falcon," https://falcon.apache.org.

[29] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems." in *USENIX ATC*, 2010.

[30] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *SIGOPS*, 2007.

[31] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *NSDI*, 2011.

[32] C. Curino, D. E. Difallah, C. Douglas, S. Krishnan, R. Ramakrishnan, and S. Rao, "Reservation-based scheduling: If you're late don't blame us!" in *SoCC*, 2014.

[33] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.

[34] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "The HaLoop approach to large-scale iterative data analysis," *The VLDB Journal*, vol. 21, no. 2, pp. 169–190, 2012.

[35] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce online," in *NSDI*, 2010.

[36] A. Rasmussen, M. Conley, G. Porter, R. Kapoor, A. Vahdat *et al.*, "Themis: an i/o-efficient mapreduce," in *SoCC*, 2012.

[37] G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in map-reduce clusters using mantri," in *OSDI*, 2010.

[38] J. Dittrich, J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, "Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing)," *Proceedings of the VLDB Endowment*, 2010.

[39] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia, "Skewtune: mitigating skew in mapreduce applications," in *SIGMOD*, 2012.

[40] Y. Wang, X. Que, W. Yu, D. Goldenberg, and D. Sehgal, "Hadoop acceleration through network levitated merge," in *SC*, 2011.

[41] K. S. Beyer and et. al, "Jaql: A scripting language for large scale semistructured data analysis," in *PVLDB*, 2011.

[42] J. Zhou, N. Bruno, M.-C. Wu, P.-A. Larson, R. Chaiken, and D. Shakib, "Scope: Parallel databases meet mapreduce," *VLDB Journal*, vol. 21, no. 5.

[43] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Disk-locality in datacenter computing considered irrelevant," in *HotOS*, 2011.

[44] B.-G. Chun and et. al., "Reef: Retainable evaluator execution framework," *PVLDB*, 2013.

[45] Twill developer community, "Twill: Apache incubator project," 2014, http://twill.incubator.apache.org/.

[46] "Apache Tez," https://tez.apache.org.

[47] "Cascading: Application platform for enterprise big data," Concurrent, Inc., 2014, http://www.cascading.org.

[48] Y. Huai and et al., "Major technical advancements in apache hive," in *SIGMOD*, 2014.

[49] "Apache Lens," https://lens.incubator.apache.org.

[50] C. Engle and et al., "Shark: fast data analysis using coarse-grained distributed memory," in *SIGMOD*, 2012.

[51] H. Choi and et al., "Tajo: A distributed data warehouse system on large clusters," in *ICDE*, 2013.

[52] "Apache Phoenix," https://phoenix.apache.org.

[53] M. Hausenblas and J. Nadeau, "Apache drill: interactive ad-hoc analysis at scale," *Big Data*, vol. 1, no. 2, pp. 100–104, 2013.

[54] "Kylin: Extreme olap engine for big data," Kylin developer community, 2014, http://kylin.io.

[55] F. M. Waas, "Beyond conventional data warehousingmassively parallel data processing with greenplum database," in *Business Intelligence for the Real-Time Enterprise*. Springer, 2009.

[56] R. D. Sloan, "A practical implementation of the data base machine-teradata dbc/1012," in *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*. IEEE, 1992.

[57] The Apache Software Foundation, "Apache Oozie: Workflow Scheduler for Hadoop," http://oozie.apache.org.

[58] "Apache Slider," https://slider.incubator.apache.org.

[59] "Hadoop Fair Scheduler," http://hadoop.apache.org/docs/r2.6.0/hadoop-yarn/hadoop-yarn-site/FairScheduler.html.

[60] "Hadoop Capacity Scheduler," http://hadoop.apache.org/docs/r2.6.0/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html.

[61] "RM should dynamically schedule containers based on utilization of currently allocated containers," https://issues.apache.org/jira/browse/YARN-1011.

[62] E. Boutin and et. al., "Apollo: scalable and coordinated scheduling for cloud-scale computing," in *OSDI*, 2014.

[63] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 2013.

[64] "Extend yarn to support distributed scheduling," https://issues.apache.org/jira/browse/YARN-2877.

[65] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *HotCloud*.

[66] F. McSherry, D. G. Murray, R. Isaacs, and M. Isard, "Differential dataflow," in *CIDR*, 2013.

[67] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *PVDLB*, 2012.

[68] Giraph developer community, "Giraph: Apache project," 2014, http://giraph.apache.org/.

[69] H. Li, A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, "Tachyon: Reliable, memory speed storage for cluster computing frameworks," in *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 2014, pp. 1–15.