Performance evaluation of Big Data analysis

Jorge Veiga, Roberto R. Expósito, and Juan Touriño

Synonyms

Big Data performance characterization

Definition

Evaluating the performance of Big Data systems is the usual way of getting information about the expected execution time of analytics applications. These applications are generally used to extract meaningful information from very large input datasets. There exist many high-level frameworks for Big Data analysis, each one oriented to different fields like machine learning and data mining, like Mahout (Apache Mahout (2009)), or graph analytics like Giraph (Avery (2011)). These high-level frameworks allow to define complex data processing pipelines that are later decomposed into more fine-grained operations in order to be executed by Big Data processing frameworks like Hadoop (Dean and Ghemawat (2008)), Spark (Zaharia et al (2016)) and Flink (Apache Flink (2014)). Therefore, the performance evaluation of these frameworks is key to determine their suitability for scalable Big Data analysis.

Big Data processing frameworks can be broken down in several layers, which typically include a data processing engine (e.g., Hadoop MapReduce), a resource manager (e.g., YARN) and a distributed storage system (e.g., HDFS). In order to provide scalability, these frameworks are deployed over the nodes of a cluster, which has a certain set of characteristics (e.g., number of nodes, CPU model, disk and network technology). Hence, the performance of Big Data systems is affected by multiple factors related both to the software components of the frameworks and the available resources in the cluster.

Most performance evaluation studies are oriented to compare several Big Data frameworks and/or different configuration alternatives. In order to do so, a set of experiments is carried out, which involves generating some input datasets, process them using several representative Big Data workloads and extract the corresponding performance metrics. The obtained results are then analyzed to find performance bottlenecks or potential optimizations.

Overview

The performance evaluation of Big Data systems typically takes into account at least one of the following metrics:

Jorge Veiga · Roberto R. Expósito · Juan Touriño
Computer Architecture Group,
Universidade da Coruña, Campus de A Coruña, 15071 A Coruña, Spain
e-mail: {jorge.veiga,rreye,juan}@udc.es

Execution time

Execution time is generally regarded as the main performance metric. It determines the wall-clock time that users have to wait for the result of their applications when executed with a particular Big Data framework.

Scalability

Scalability is the ability of a Big Data system to increase its performance when adding more resources. Two different kinds of scalability can be considered, vertical scalability (scaling up) and horizontal scalability (scaling out). On the one hand, vertical scalability involves obtaining faster nodes with more powerful processors and more memory. On the other hand, horizontal scalability involves adding more nodes to the system and operate in a distributed environment.

Resource utilization

The leveraging of system resources (e.g., CPU, network, disk) determines the adaptability of the frameworks to a particular system. Performance bottlenecks can be due to underutilization or overloading of these resources.

Energy efficiency

Performance is closely related to energy efficiency, as the energy consumed when executing a workload is determined by the power specifications of the system and the execution time. Modifying the underlying system by increasing the cluster size or the computational capabilities of the nodes can reduce execution time, but it may increase the power consumption of the workload. Therefore, determining whether an optimization can improve energy efficiency requires a thorough analysis.

Microarchitectural behavior

Accessing hardware performance counters that are present in modern processors, like the number of instructions executed or cache misses, can provide meaningful information to characterize the interaction between frameworks and hardware. Moreover, these metrics can be utilized to compare different alternatives in a more fine-grained fashion than just considering overall metrics like execution time. The values of the counters can be accessed in several ways, using APIs like PAPI (Browne et al (2000)) or monitoring tools like perf and Oprofile.

Key research findings

Many works have assessed and optimized the performance of Big Data systems taking into account different factors. A summary of the main results obtained is provided next.

Data processing engine

A crucial factor for the performance of Big Data frameworks is the underlying data processing engine, which defines the kind of operations that the user can perform to process the input

dataset. MapReduce has been one of the most popular batch engines so far, and Hadoop is its de-facto standard implementation. However, Hadoop presents some performance overheads, like the writing of intermediate results to disk, which has led to the development of different alternatives that optimize its performance. They whether modify some of its components, like NativeTask (Yang et al (2013)), or redesign the entire underlying architecture, like Flame-MR (Veiga et al (2016c)). Although these works significantly improve the performance of Hadoop, they are still limited by the nature of the MapReduce model.

More advanced in-memory frameworks like Spark and Flink are designed to provide a wider range of data operators than MapReduce, as well as support for other scenarios (e.g., real-time/streaming processing). Although they support map and reduce functions, existing MapReduce applications must be adapted to their new programming model. The increased flexibility of Spark and Flink along with the caching of intermediate results in memory can reduce Hadoop execution times by 77% and 70% on average, respectively (Veiga et al (2016a)).

Other works have explored the possibilities offered by paradigms traditionally oriented to High Performance Computing (HPC). For example, parallel programming paradigms like the Message Passing Interface (MPI) can increase significantly the performance of Big Data workloads (Liang et al (2014); González et al (2017)). However, MPI offers a low-level API that provides poor programming productivity compared to MapReduce, so its use is not feasible for real Big Data scenarios.

File system

Big Data frameworks typically use the Hadoop Distributed File System (HDFS) to distribute the storage of large datasets over the nodes of a cluster, collocating storage and compute services on the same nodes. However, its use is not widespread in HPC systems, which separate compute and storage services by using parallel file systems like GPFS, OrangeFS or Lustre. This situation has caused the appearance of several works that evaluate the performance of both storage approaches, concluding that GPFS behaves better at low concurrency, while HDFS is more suited to high concurrency (Fadika et al (2012)). Some other works have shown that parallel file systems can also provide performance improvements, like MARIANE (Fadika et al (2014)), which uses a custom MapReduce implementation onto GPFS. Another approach presented by Xuan et al (2017) uses a two-level storage system by integrating the in-memory file system Tachyon with OrangeFS to obtain higher performance than just using HDFS.

Disk technology

Traditional Hard Disk Drives (HDDs) are progressively being replaced by faster technologies like Solid-State Drives (SSDs), which obtain significantly better performance but at higher cost per byte. Using SSDs has been reported to improve the performance of Big Data frameworks. Hadoop can store HDFS data and intermediate results on SSDs to eliminate disk bottlenecks when executing I/O-bound workloads. However, SSD disks can increase dra-

matically the total cost of the system components (Moon et al (2014)). Regarding Spark, its performance can be improved by 23% when using SSDs to store intermediate results compared to the memory-only approach (Choi et al (2015)).

SSDs can also be leveraged actively, performing some simple operations over the stored data. In Lee et al (2016), a new technique called external sorting utilizes SSDs to perform sort operations during the MapReduce phase, reducing the execution time of Hadoop by up to 36%.

Network interconnects

Early works claimed that the use of high performance interconnects like InfiniBand would not have a great impact on the execution time of MapReduce workloads, unless the shuffle algorithm and communication protocol were modified (Fadika et al (2012)). However, more modern evaluations have shown that up-to-date Hadoop versions can leverage this kind of networks by using IP over InfiniBand (IPoIB), obtaining significant performance improvements (Veiga et al (2016b)).

Other works modify the shuffle components to take advantage of Remote Direct Memory Access (RDMA) communications. That is the case of the network levitated merge algorithm (Wang et al (2011)) and RDMA-Hadoop (Wasi-Ur-Rahman et al (2013)). The latter claims to reduce the execution time of Hadoop and the network levitated merge algorithm by 32% and 21%, respectively, for the TeraSort benchmark. The use of RDMA has also been studied for Spark, showing up to 46% performance improvement for several workloads (Lu et al (2016b)).

Memory management

As Big Data applications read and generate a lot of data, managing the available memory resources correctly is key to achieve good performance and avoid memory overflows. Most Big Data frameworks are written in some managed language (e.g., Java, Scala) executed by the Java Virtual Machine (JVM). In this context, objects are tracked to release their memory once they stop being referenced. This process is performed by the garbage collector and can cause significant performance overheads when processing large datasets.

Modifying the original JVM memory management to adapt it to the characteristics of Big Data systems can lead to significant performance improvement. That is shown by proposals like Broom (Gog et al (2015)), which uses a region-based algorithm to allocate data objects. Another example, Yak (Nguyen et al (2016)), implements a hybrid approach that utilizes generation-based and region-based algorithms for control and data objects, respectively. As these memory managers are implemented in Java, they can be generically applied to any JVM-based framework.

Other solutions are specific to a certain framework, like Deca (Lu et al (2016a)), which modifies the management of data containers in Spark to estimate their lifetime, allocating memory regions accordingly. This mechanism, combined with other optimizations like the use of byte arrays to store data objects, is able to achieve

up to 41.6× speedup in cases with data spilling.

Manycore accelerators

The great majority of Big Data frameworks rely only on CPUs to perform the computations. However, some works propose the use of manycore accelerators, typically available in heterogeneous systems, like GPUs, FPGAs or Xeon Phi accelerators.

GPUs are a suitable option to accelerate Big Data workloads due to their widespread use and high degree of data parallelism. For example, Mars (Fang et al (2011)) supports the processing of MapReduce workloads using CPU, GPU or hybrid computations. The combination of Hadoop with Mars can provide a maximum speedup of 2.8. GPUs have also been employed to improve the performance of Spark (Yuan et al (2016)) and Flink (Chen et al (2017)).

Another popular type of accelerator is the Xeon Phi manycore processor that can execute unmodified CPU code. However, the source code of Big Data frameworks must be adapted in order to fully leverage the computational power of this accelerator, as presented by Lu et al (2015).

Finally, FPGAs are hardware devices that can be programmed to build custom accelerators. Neshatpour et al (2015) assess the benefits of accelerating typical machine learning and data mining applications by offloading some of their kernels to FPGAs. They obtain a speedup of 2.72, although it must be taken into account that the accelerated framework is highly application-specific.

System architecture

Some studies have evaluated the performance of "big" Intel Xeon nodes compared to the use of "small" nodes like ARM or Intel Atom. Loghin et al (2015) state that "big" nodes are more efficient for CPU-intensive jobs, while "small" ones perform better for I/O-intensive workloads. Malik et al (2015) conclude that "big" nodes are more efficient as the computational size of the problem increases.

Other evaluations focus on determining whether horizontal scalability is more beneficial than vertical scalability. The results are highly dependent on the framework used and the workload characterization. In general terms, horizontal scalability provides better performance for Hadoop (Li and Shen (2017)), while Spark presents better energy efficiency when using vertical scalability (Yoo et al (2016)).

Examples of application

A wide range of tools have been developed to evaluate the performance of Big Data systems and frameworks. Most of them are benchmark suites that provide multiple kinds of workloads to assess the performance of different use cases. Although the great majority is oriented to Hadoop, like HiBench (Huang et al (2010)) or BigDataBench (Wang et al (2014)), some new ones target in-memory frameworks, like SparkBench (Li et al (2017)) for Spark.

Other evaluation tools not only provide a set of benchmarks but also ease the execution of the experiments. That is the case of MRBS (Sangroya et al

(2012)), which is able to automatically set up a Hadoop cluster in a public cloud provider. Once the cluster is running, it injects the dataset and executes the workloads, obtaining execution time, throughput and cost metrics.

BDEv (formerly MREv, Veiga et al (2015)) is another evaluation tool that supports several flavors of Hadoop, Spark and Flink. Once the user configures a set of experimental parameters, it launches the frameworks, generates the input datasets and performs the experiments. It also automatically records several metrics, including execution time, resource utilization, energy efficiency and hardware performance counters.

Big Data Watchdog (BDWatchdog) (Enes et al (2017)) is another tool that enables to record resource utilization statistics for the individual processes involved in the execution of the frameworks (e.g., DataNode, Spark Executor). Moreover, it can provide real-time profiling information about the execution of the JVM code.

Future directions for research

As explained in previous sections, performance is affected by multiple factors. Although different studies have addressed the performance evaluation of Big Data analysis, end users rarely benefit from the research findings provided by these studies. There is still work to be done regarding the development of tools that can bring more meaningful insights to users.

Users that want to select a Big Data framework to utilize in a certain infrastructure can compare their options by performing several experiments. This involves the deployment of several frameworks (e.g., Hadoop, Spark) and testing their performance. In addition to the effort associated to this evaluation, the optimal choice may still require a more thorough analysis, taking also into account different classes of workloads that the user may be willing to execute (e.g., CPU-intensive, I/O-intensive) and varying the configuration parameters of the frameworks (e.g., maps per node, executors per node). Furthermore, some information must be given to the user regarding the performance bottlenecks that may exist, and which system resource may be the best option for improving.

Although there are some evaluation tools that ease the execution of these tasks, future ones must incorporate some knowledge to help users to make decisions, not only retrieving raw performance information.

References

Apache Flink (2014) Scalable batch and stream data processing. http://flink.apache.org/, [Last visited: December 2017]

Apache Mahout (2009) Scalable machine learning and data mining. http://mahout.apache.org/, [Last visited: December 2017]

Avery C (2011) Giraph: large-scale graph processing infrastructure on Hadoop. In: 2011 Hadoop Summit, Santa Clara, CA, USA, pp 5–9

Browne S, Dongarra J, Garner N, Ho G, Mucci P (2000) A portable programming interface for performance evaluation on modern processors. The International Journal of High Performance Computing Applications 14(3):189–204

Chen C, Li K, Ouyang A, Tang Z, Li K (2017) GPU-accelerated parallel hierarchical ex-

treme learning machine on Flink for Big Data. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47(10):2740–2753

Choi IS, Yang W, Kee YS (2015) Early experience with optimizing I/O performance using high-performance SSDs for in-memory cluster computing. In: 2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, pp 1073–1083

Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Communications of the ACM 51(1):107–113

Enes J, Expósito RR, Touriño J (2017) Big Data Watchdog: real-time monitoring and profiling. `http://bdwatchdog.dec.udc.es`, [Last visited: December 2017]

Fadika Z, Govindaraju M, Canon R, Ramakrishnan L (2012) Evaluating Hadoop for data-intensive scientific operations. In: 5th IEEE International Conference on Cloud Computing (CLOUD'12), Honolulu, HI, USA, pp 67–74

Fadika Z, Dede E, Govindaraju M, Ramakrishnan L (2014) MARIANE: using MApReduce in HPC environments. Future Generation Computer Systems 36:379–388

Fang W, He B, Luo Q, Govindaraju NK (2011) Mars: accelerating MapReduce with graphics processors. IEEE Transactions on Parallel and Distributed Systems 22(4):608–620

Gog I, Giceva J, Schwarzkopf M, Vaswani K, Vytiniotis D, Ramalingan G, Costa M, Murray D, Hand S, Isard M (2015) Broom: sweeping out garbage collection from Big Data systems. In: 15th Workshop on Hot Topics in Operating Systems (HotOS'15), Kartause Ittingen, Switzerland

González P, Pardo XC, Penas DR, Teijeiro D, Banga JR, Doallo R (2017) Using the cloud for parameter estimation problems: comparing Spark vs MPI with a case-study. In: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017), Madrid, Spain, pp 797–806

Huang S, Huang J, Dai J, Xie T, Huang B (2010) The HiBench benchmark suite: characterization of the MapReduce-based data analysis. In: 26th IEEE International Conference on Data Engineering Workshops (ICDEW'10), Long Beach, CA, USA, pp 41–51

Lee YS, Quero LC, Kim SH, Kim JS, Maeng S (2016) ActiveSort: efficient external sorting using active SSDs in the MapReduce framework. Future Generation Computer Systems 65:76–89

Li M, Tan J, Wang Y, Zhang L, Salapura V (2017) SparkBench: a Spark benchmarking suite characterizing large-scale in-memory data analytics. Cluster Computing 20(3):2575–2589

Li Z, Shen H (2017) Measuring scale-up and scale-out Hadoop with remote and local file systems and selecting the best platform. IEEE Transactions on Parallel and Distributed Systems 28(11):3201–3214

Liang F, Feng C, Lu X, Xu Z (2014) Performance benefits of DataMPI: a case study with BigDataBench. In: 4th Workshop on Big Data Benchmarks, Performance Optimization and Emerging Hardware (BPOE'14), Salt Lake City, UT, USA, pp 111–123

Loghin D, Tudor BM, Zhang H, Ooi BC, Teo YM (2015) A performance study of Big Data on small nodes. Proceedings of the VLDB Endowment 8(7):762–773

Lu L, Shi X, Zhou Y, Zhang X, Jin H, Pei C, He L, Geng Y (2016a) Lifetime-based memory management for distributed data processing systems. Proceedings of the VLDB Endowment 9(12):936–947

Lu M, Liang Y, Huynh HP, Ong Z, He B, Goh RSM (2015) MrPhi: an optimized MapReduce framework on Intel Xeon Phi coprocessors. IEEE Transactions on Parallel and Distributed Systems 26(11):3066–3078

Lu X, Shankar D, Gugnani S, Panda DK (2016b) High-performance design of Apache Spark with RDMA and its benefits on various workloads. In: 2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington, DC, USA, pp 253–262

Malik M, Rafatirah S, Sasan A, Homayoun H (2015) System and architecture level characterization of Big Data applications on big and little core server architectures. In: 2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, pp 85–94

Moon S, Lee J, Kee YS (2014) Introducing SSDs to the Hadoop MapReduce framework. In: 7th IEEE International Confer-

ence on Cloud Computing (CLOUD'14), Anchorage, AK, USA, pp 272–279

Neshatpour K, Malik M, Ghodrat MA, Sasan A, Homayoun H (2015) Energy-efficient acceleration of Big Data analytics applications using FPGAs. In: 2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, pp 115–123

Nguyen K, Fang L, Xu GH, Demsky B, Lu S, Alamian S, Mutlu O (2016) Yak: a high-performance Big-Data-friendly garbage collector. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16), Savannah, GA, USA, pp 349–365

Sangroya A, Serrano D, Bouchenak S (2012) MRBS: towards dependability benchmarking for Hadoop MapReduce. In: 18th International Euro-Par Conference on Parallel Processing Workshops (Euro-Par'12), Rhodes Island, Greece, pp 3–12

Veiga J, Expósito RR, Taboada GL, Touriño J (2015) MREv: an automatic MapReduce Evaluation tool for Big Data workloads. In: International Conference on Computational Science (ICCS'15), Reykjavík, Iceland, pp 80–89

Veiga J, Expósito RR, Pardo XC, Taboada GL, Touriño J (2016a) Performance evaluation of Big Data frameworks for large-scale data analytics. In: 2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington, DC, USA, pp 424–431

Veiga J, Expósito RR, Taboada GL, Touriño J (2016b) Analysis and evaluation of MapReduce solutions on an HPC cluster. Computers & Electrical Engineering 50:200–216

Veiga J, Expósito RR, Taboada GL, Touriño J (2016c) Flame-MR: an event-driven architecture for MapReduce applications. Future Generation Computer Systems 65:46–56

Wang L, Zhan J, Luo C, Zhu Y, Yang Q, He Y, Gao W, Jia Z, Shi Y, Zhang S, Zheng C, Lu G, Zhan K, Li X, Qiu B (2014) BigDataBench: a Big Data benchmark suite from Internet services. In: 20th IEEE International Symposium on High-Performance Computer Architecture (HPCA'14), Orlando, FL, USA, pp 488–499

Wang Y, Que X, Yu W, Goldenberg D, Sehgal D (2011) Hadoop acceleration through network levitated merge. In: International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), Seattle, WA, USA, pp 57:1–57:10

Wasi-Ur-Rahman M, Islam NS, Lu X, Jose J, Subramoni H, Wang H, Panda DK (2013) High-performance RDMA-based design of Hadoop MapReduce over InfiniBand. In: 27th IEEE International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW'13), Boston, MA, USA, pp 1908–1917

Xuan P, Ligon WB, Srimani PK, Ge R, Luo F (2017) Accelerating Big Data analytics on HPC clusters using two-level storage. Parallel Computing 61:18–34

Yang D, Zhong X, Yan D, Dai F, Yin X, Lian C, Zhu Z, Jiang W, Wu G (2013) NativeTask: a Hadoop compatible framework for high performance. In: 2013 IEEE International Conference on Big Data (IEEE BigData 2013), Santa Clara, CA, USA, pp 94–101

Yoo T, Yim M, Jeong I, Lee Y, Chun ST (2016) Performance evaluation of in-memory computing on scale-up and scale-out cluster. In: 8th International Conference on Ubiquitous and Future Networks (ICUFN 2016), Vienna, Austria, pp 456–461

Yuan Y, Salmi MF, Huai Y, Wang K, Lee R, Zhang X (2016) Spark-GPU: an accelerated in-memory data processing engine on clusters. In: 2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington, DC, USA, pp 273–283

Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, Meng X, Rosen J, Venkataraman S, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I (2016) Apache Spark: a unified engine for Big Data processing. Communications of the ACM 59(11):56–65

Cross-References

- Big Data benchmarking
- Energy efficiency in Big Data analysis