# MapReduce in Computational Biology - A Synopsis

Giuseppe Cattaneo[1], Raffaele Giancarlo[2], Stefano Piotto[3], Umberto Ferraro Petrillo[4], Gianluca Roscigno[1]⋆, and Luigi Di Biasi[3]

[1] Dipartimento di Informatica
Università degli Studi di Salerno, I-84084, Fisciano (SA), Italy
`{cattaneo,giroscigno}@unisa.it`
[2] Dipartimento di Matematica ed Informatica
Università di Palermo, I-90133, Palermo (PA), Italy
`raffaele.giancarlo@unipa.it`
[3] Dipartimento di Farmacia
Università degli Studi di Salerno, I-84084, Fisciano (SA), Italy
`piotto@unisa.it`
[4] Dipartimento di Scienze Statistiche
Università di Roma *"La Sapienza"*, I-00185, Roma, Italy
`umberto.ferraro@uniroma1.it`

**Abstract.** In the past 20 years, the Life Sciences have witnessed a paradigm shift in the way research is performed. Indeed, the computational part of biological and clinical studies has become central or is becoming so. Correspondingly, the amount of data that one needs to process, compare and analyze, has experienced an exponential growth. As a consequence, High Performance Computing (HPC, for short) is being used intensively, in particular in terms of multi-core architectures. However, recently and thanks to the advances in the processing of other scientific and commercial data, Distributed Computing is also being considered for Bioinformatics applications. In particular, the MapReduce paradigm, together with the main middleware supporting it, i.e., Hadoop and Spark, is becoming increasingly popular.
Here we provide a short review in which the state of the art of MapReduce bioinformatics applications is presented, together with a qualitative evaluation of each of the software systems that have been here included. In order to make the paper self-contained, computer architectural and middleware issues are also briefly presented.
**Keywords:** Bioinformatics, Distributed Computing, MapReduce, Hadoop, Spark.

## 1 Introduction

The development of sequencing technologies has caused a stunning reduction in sequencing costs, well illustrated in Figure 1, whose effect is an exponential

⋆ Corresponding author: Gianluca Roscigno, `giroscigno@unisa.it`, Phone: +39089969304 increase in the production of genomic and proteomic sequences that need to be analyzed. Unfortunately, computer hardware costs have not kept the same reduction pace, causing an economic problem to research areas that use sequencing, i.e., the entire Life Sciences. Those aspects are well summarized in [1,2], where it is envisioned that the cost of analyzing sequence data may be a factor of 100 times more than the cost of its production. Solutions are also proposed, one of which is to use Cloud and Distributed Computer Systems to support the computer-related aspects of research in the Life Sciences. Although *High Performance Computing* (HPC) is a fundamental part of Bioinformatics and Computational Biology, it has been mainly used for computationally hard problems such as prediction of protein structure. Unfortunately, the scenario has changed dramatically due to the simple sheer quantity of data to be processed: even the simple task of counting the number of $k$-mers in a set of strings, a fundamental problem in Genomics and Epigenomics (see, e.g., [3,4,5]), is being addressed with the use of multi-processor architectures, e.g., [6]. As opposed to multi-processor architectures, the use of Distributed Architectures is emerging only now and it seems to be the future, due to the scalability that this type of architecture grants. In order to foster further development of Bioinformatics and Computational Biology distributed software systems and platforms, here a short review of this area is provided. Moreover, we also identify some desirable properties that those software systems should have and provide a corresponding evaluation of them. Specifically, Section 2 is dedicated to a short description of distributed computing in the management of Big Data, with emphasis on the architectural and middleware issues. In Section 3, we focus on MapReduce implementations in bioinformatics. Finally, in Section 4, some conclusions are offered.

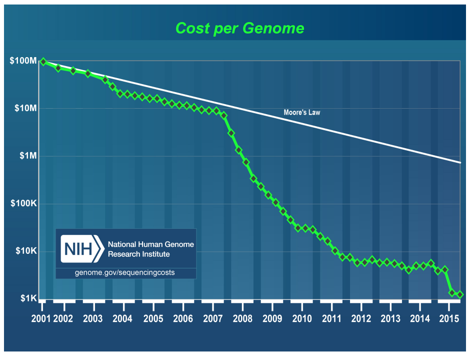

**Fig. 1.** Cost (on a logarithmic scale) to obtain a good draft of the Human Genome, as sequencing technology improves over the years. The well known Moore's law governing computer hardware costs is also shown. In 2008, it became cheaper to produce a DNA sequence with respect to the hardware that would be needed to analyze and store it. The figure is taken from [7]

## 2 Distributed Computing in the Management of Big Data: Architectural and Middleware Issues

Many bioinformatics problems can be solved by partitioning the input in a number of independent parts that can be processed independently. Therefore, many time-consuming applications in bioinformatics have been expressly designed to exploit parallelism of the multi-core shared-memory architectures. Due to a number of architectural bottlenecks mainly related to the need of multiple cores to share the same memory or I/O buses, multi-core shared-memory architectures do not allow to efficiently use more than a rather limited number of cores. For this reason, the performance of a parallel algorithm developed for those architectures may not scale when the number of cores goes beyond a certain threshold. An application can be considered *scalable* when its execution time is proportionally reduced when the number of computing units is increased.

The above shortcoming of multi-core shared-memory architectures is bypassed by resorting to *Distributed Systems*. It is well-know that this term refers to a collection of independent computers (also called *nodes*) that communicate over a network. The interested reader can find an introduction in this subject in [8]. Theoretically, with this approach, an arbitrary large number of nodes can be added to the system by exploiting its inherent scalability (i.e., *scale out*). This is possible since each node can access local resources without competing with other nodes. This feature can radically improve the performance of any properly designed application reducing its execution time.

As a drawback, the adoption of a distributed approach often requires more specific and complex skills to design and develop a distributed algorithm, given the target architecture. To promote the transition toward distributed systems, many new programming paradigms have been proposed in recent years. Among those, MapReduce [9] paradigm is becoming a *de facto* standard. It is described next, together with the two main middleware systems supporting it, i.e., Apache Hadoop [10] and Apache Spark [11,12].

### 2.1 MapReduce Paradigm

It is based on the proper definition of two functions: *map* and *reduce*. Assuming that the input is organized as a set of $<key, value>$ pairs, the generic map function takes as input one of these pairs and returns, as output, a set of intermediate $<key, value>$ pairs. The reduce function is then used to process all the intermediate pairs having the same *key*, typically returning a synoptic rendering of the input. Map and reduce functions are executed, as tasks, by workers running on the nodes of a distributed system complying to a MapReduce framework. The communication between workers running map functions and workers running reduce functions is accomplished in an automatic and transparent way by the underlying framework (*implicit parallelism*). Therefore, the programmer can focus on the definition of the map and reduce functions, since all the aspects related to the execution in a distributed setting (e.g., the number of concurrent map and reduce tasks to issue) are addressed via the proper definition of configuration variables.

An overview of MapReduce paradigm is depicted in Figure 2.

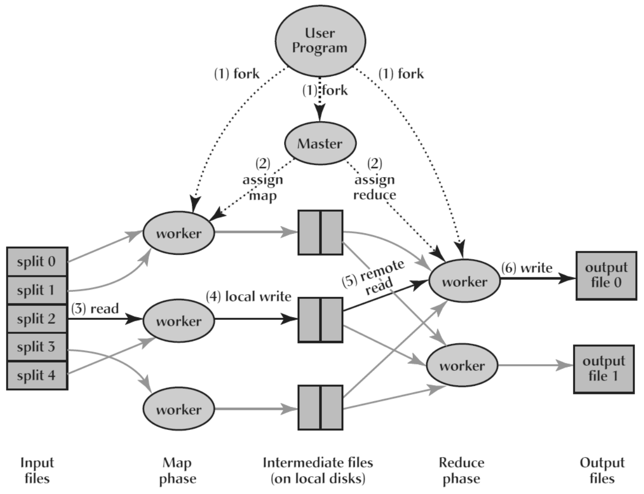

**Fig. 2.** Workflow of a MapReduce application (taken from [9]). Input files are split in several parts, with each part processed by a distinct map task according to a user-defined function. Each of these tasks emits, as output, 0, 1 or more intermediate $<K, V>$ pairs. Then, these pairs are shuffled and sorted so that all the pairs having the same key are processed by the user-defined function of a same reduce task. These produce, as output, a set of $<K, V>$ pairs that are appended to a different part of the same logical file.

**Apache Hadoop** It is currently the most popular and mature framework supporting MapReduce. It is mainly composed of two modules: YARN (*Yet Another Resource Negotiator*) [13] and HDFS (*Hadoop Distributed File System*) [14]. YARN is a data processing framework supporting the execution of distributed algorithms through different types of computing paradigms. HDFS is a distributed and block-structured file system designed to run on commodity hardware and able to provide fault tolerance through replication of data.

In general, a Hadoop cluster consists of a single *master node* and multiple *slave nodes*: the master node runs the `Resource Manager` and the `Name Node` services, while slave nodes run the `Data Node` and the `Node Manager` services. On the master node, the `Resource Manager` arbitrates the assignment of computational resources to applications. On the slave nodes, the `Node Manager` monitors and keeps informed the `Resource Manager` about the status of the node. Again, on the master node, the `Name Node` service maintains the directory tree of all files existing in the HDFS and keeps tracks of where data blocks are physically placed. On the slave nodes, the `Data Node` service maintains a subset of the HDFS data blocks using the local storage.

Applications are run on Hadoop via an `Application Master`. This is a service instantiated by a `Node Manager` on a slave node upon a request coming from the `Resource Manager`. Once created, it asks the Hadoop framework for all the resources required to perform a computation (mainly in terms of CPU and memory). The `Resource Manager` responds by reserving to the application a set of *worker*s (also called `Containers`) running on one or more slave nodes (a worker is the basic processing unit in Hadoop to execute map or reduce tasks).

*HDFS: Hadoop Distributed File System* In a distributed system it is often more efficient to run a task on local data, rather than to move the data where the task is running. In fact, one of the main characteristics of Hadoop is its ability to exploit data local computing. In particular, HDFS provides functionality to enable applications to move closer to the data, minimizing network congestion and increasing the overall throughput of the system.

It is a distributed and block-structured file system, optimized to run also on commodity hardware and able to provide fault tolerance through replication of data. The HDFS is able to reliably maintain very large files, in fact, it works by automatically splitting large files in smaller blocks and spreading them across nodes in a network.

*Lifetime of a Hadoop MapReduce Algorithm* The execution of a Hadoop MapReduce application takes place in two consecutive (and potentially overlapping) phases: the map phase and the reduce phase. During the map phase, one or more map tasks are run by workers on the slave nodes of the Hadoop cluster. Each worker may run one task a time, while several workers may run in parallel on the same slave node. When an application is running, a worker in the cluster is dedicated to the `Application Master` service.

The execution of a map task goes through four phases. At startup, the task initializes the data structures required for managing the input and the output of the task (*init* phase). Then, each worker begins the execution of the map functions (*execution* phase). As soon as output pairs are returned, these are saved in a temporary memory buffer. When the buffer gets almost full or when the map functions execution ends, the output pairs are sorted, partitioned according to the destination reduce task and written on disk (*spilling* phase). Then, data belonging to each partition are moved to the slave nodes where the reduce tasks will process them.

The execution of a reduce task requires three phases. At the beginning, all the pairs produced by map tasks and included in a certain partition are moved on the node where the reduce tasks assigned to that partition will be run (*shuffle* phase). As soon as new pairs are received by a node, they are sorted in order to keep them grouped according to their key (*sort* phase). Finally, for each group of pairs with the same key, a reduce function will be run by a worker on that node.

**Apache Spark** It is a distributed framework that supports applications with acyclic data-flow model and in-memory computing. This framework also preserves the scalability and fault tolerance of MapReduce-compliant frameworks. Spark provides APIs in Scala, Java and Python programming languages, and it can be used for programs that reuse a working set of data across multiple parallel operations, such as iterative algorithms.

Apache Spark also has a *master/slaves* architecture like Hadoop. In particular, there are two main services: `Worker` and `Driver`. A `Worker` is a service running on any slave node that can execute application code. An `Executor` is a process launched for an application on a `Worker` node that runs tasks and keeps data in memory or disk storage. In addition, each application has its own `Executor`s.

The `Driver` service runs on the master node and it manages the `Executor`s through a cluster resource manager, e.g., stand-alone cluster manager of Spark, Apache Mesos or Hadoop YARN. In particular, the `Worker`s create `Executor`s for the `Driver`, and then it can run tasks in those `Executor`s.

The main feature of Spark is the *Resilient Distributed Dataset* (RDD). It represents a read-only collection of objects partitioned across a set of nodes that can be rebuilt if a partition is lost. An RDD can be used to cache data in memory across nodes and it can be reused in multiple MapReduce-based operations. Several parallel operations can be performed on RDDs, such as: *reduce*, *collect* and *foreach*. The *reduce* operation combines dataset elements using an associative function to produce a result, while the *collect* operation sends all elements of the dataset to the `Driver`. Instead, the *foreach* operation passes each element through a user provided function.

In addition, Spark also provides `Dataset`s and `DataFrame`s features. The first is a distributed collection of data providing an interface that combines the benefits of RDDs with those of Spark SQL's optimized execution engine. A `Dataset` can be constructed from objects and then manipulated using functional transformations, such as: *map*, *flatMap*, *filter*, etc. Instead, a Spark `DataFrame` is a `Dataset` organized into named columns, and it is equivalent to a table concept in a relational database, but providing more optimizations. Roughly speaking, a `DataFrame` is represented by a `Dataset` of rows.

Apache Spark can have the Hadoop framework as the underlining middleware. Moreover, as opposed to Hadoop, it is not limited to support the MapReduce paradigm. Finally, it allows the combination of streaming and batch processing, while Hadoop can be only used for batch applications.

## 3 State of Art of MapReduce-based Software in Bioinformatics

In this section, we report a classification of bioinformatics software tools that are based on the MapReduce paradigm. The vast majority of them is supported by Hadoop and some by Spark. A new special purpose framework has also been proposed in [15], i.e., GATK.

As far as the qualitative evaluation of the presented software, the properties that are desirable each must have are the following.

(a) **Programming Platform (PP)** This property means that the software has been designed as a flexible programming platform to be used for a set of customizable analyses. This property is held by any application that allows custom queries or high level programming interface (API) to develop new applications. For example the Hadoop-based BioPig [16] tool is considered a good platform to solve bioinformatics problems and, therefore, holds this property.

(b) **Booster** In some specific cases and in order to gain in time performance, MapReduce applications are developed by distributing multiple instances of a sequential package/library already existing, which is used as black box. Therefore, such a wrapping provides a boosting of the sequential package. For instance, CloudBLAST [17] has been developed starting from a well-known existing application (namely BLAST [18]) through a "wrapper" for the Hadoop framework. Another example is GRIMD [19] that was built on Microsoft Windows Server System and on Microsoft VPN PT2P protocol to set up secure connections between clients and server. GRIMD is fully configurable and permits the deploy of quantum mechanical [20] as well as molecular dynamics calculations [21,22] and genome analysis.

(c) **New Algorithm (NewA)** In this case the MapReduce application is developed based on a new algorithm (expressly designed for a target distributed architecture) to solve a classic or new bioinformatics problem. For instance, CloudAligner [23] has been specifically designed according to the MapReduce paradigm and then implemented in Hadoop.

(d) **New Algorithm and Engineering (NewAE)** This property holds if the MapReduce application is developed adopting an algorithm engineering approach [24,25] which, in addition to property (c), is supposed to provide highly tuned code.

The software can be divided into the following categories.

- ***Tools and programming environments for the development of Bioinformatics applications***. We have included in this category programming tools and environments used to develop MapReduce pipelines and programs for bioinformatics (see Table 1), e.g., processing of HTS sequence data. In addition, we have also included libraries that provide interfaces that allow high level applications to work on files that have a standard format in bioinformatics, such as Hadoop-BAM [26].
- ***Algorithms for Single Nucleotide Polymorphism Identification***. We have included in this category software that performs SNP identification and analysis (see Table 2).
- ***Gene Expression Analysis***. We have included in this category software for gene expression analysis (see Table 3), e.g., gene set analysis for biomarker identification.

– ***Sequence Comparison***. We have included in this category software for sequence comparison based on alignments and alignment-free methods (see Table 4).
– ***Genome Assembly***. We have included in this category software for *de novo* genome assembly from short sequencing reads (see Table 5).
– ***Sequencing Reads Mapping***. We have included in this category software for mapping short reads to reference genomes (see Table 6).
– ***Additional Applications***. We have included in this category MapReduce bioinformatics applications for which there is only one implementation available (see Table 7).

**Table 1.** *Tools and programming environments for the development of Bioinformatics applications*. With reference to the main text, the software reported in this category is classified as shown. YES indicates that the property is held, while NO indicates otherwise. Other possible values indicate to what extent the property is held: GE stands for "to a great extent", ME stands for "to a moderate extent" and SE stands for "to a small extent".

| **Software** | **PP** | **Booster** | **NewA** | **NewAE** |
|---|---|---|---|---|
| **BioPig** [16] | YES | YES | GE | NO |
| **Cloudgene** [27] | YES | NO | NO | NO |
| **FASTdoop** [28] | YES | NO | GE | GE |
| **GATK** [15] | YES | NO | GE | NO |
| **Hadoop-BAM** [26] | YES | NO | GE | NO |
| **SeqPig** [29] | YES | YES | GE | NO |
| **SparkSeq** [30] | YES | NO | SE | SE |

**Table 2.** *Algorithms for Single Nucleotide Polymorphism Identification*. This table is analogous to Table 1

| **Software** | **PP** | **Booster** | **NewA** | **NewAE** |
|---|---|---|---|---|
| **BlueSNP** [31] | YES | YES | NO | NO |
| **Crossbow** [32] | NO | YES | SE | NO |

## 4 Conclusion

We have presented the state of the art regarding the use of Distributed Computing, in particular software designed with MapReduce paradigm, in Bioinformatics and Computational Biology. Although such a body of work will grow in the

**Table 3.** *Gene Expression Analysis.* This table is analogous to Table 1

| Software | PP | Booster | NewA | NewAE |
|---|---|---|---|---|
| **Eoulsan** [33] | NO | YES | NO | NO |
| **FX** [34] | NO | NO | GE | NO |
| **MyRNA** [35] | YES | YES | SE | SE |
| **YunBe** [36] | NO | NO | GE | NO |

**Table 4.** *Sequence Comparison.* This table is analogous to Table 1

| Software | PP | Booster | NewA | NewAE |
|---|---|---|---|---|
| ***Almeida et al.*** [37] | NO | NO | GE | NO |
| **CloudBLAST** [17] | NO | YES | NO | NO |
| **HAFS** [38] | YES | NO | GE | GE |
| **K-mulus** [39] | NO | YES | NO | NO |
| **Nephele** [40] | NO | NO | NO | NO |
| **Strand** [41] | NO | NO | GE | NO |

**Table 5.** *Genome Assembly.* This table is analogous to Table 1

| Software | PP | Booster | NewA | NewAE |
|---|---|---|---|---|
| **CloudBrush** [42] | NO | NO | GE | NO |
| **Contrail** [43] | NO | YES | GE | NO |

**Table 6.** *Sequencing Reads Mapping.* This table is analogous to Table 1

| Software | PP | Booster | NewA | NewAE |
|---|---|---|---|---|
| **BlastReduce** [44] | NO | NO | GE | NO |
| **CloudAligner** [23] | NO | NO | GE | NO |
| **CloudBurst** [45] | NO | NO | GE | NO |
| **SEAL** [46] | NO | YES | NO | NO |
| **SparkSW** [47] | NO | NO | GE | ME |

**Table 7.** *Additional Applications.* This table is analogous to Table 1

| Software | PP | Booster | NewA | NewAE |
|---|---|---|---|---|
| **BioDoop** [48] | NO | YES | SE | SE |
| ***Codon Counting*** [49] | NO | NO | GE | NO |
| **GRIMD** [19] | YES | YES | ME | ME |
| **MrMC-MinH** [50] | NO | NO | GE | NO |
| **MrsRF** [51] | NO | NO | GE | ME |
| **PeakRanger** [52] | NO | NO | GE | NO |

future, it would be appropriate to follow good design and algorithm engineering approaches in order to use in full the available hardware and the scalability MapReduce offers.

## References

1. Kahn, S.D.: On the future of genomic data. Science 331, 728–729 (2011)
2. Mardis, E.R.: The $1,000 genome, the $100,000 analysis? Genome Medicine 2, 1–3 (2010)
3. Compeau, P.E.C., Pevzner, P.A., Tesler, G.: How to apply de Bruijn graphs to genome assembly. Nature Biotechnology 29, 987–991 (2011)
4. Giancarlo, R., Rombo, S.E., Utro, F.: Epigenomic k-mer dictionaries: shedding light on how sequence composition influences in vivo nucleosome positioning. Bioinformatics (2015)
5. Utro, F., Di Benedetto, V., Corona, D.F., Giancarlo, R.: The intrinsic combinatorial organization and information theoretic content of a sequence are correlated to the DNA encoded nucleosome organization of eukaryotic genomes. Bioinformatics (2015)
6. Deorowicz, S., Kokot, M., Grabowski, S., Debudaj-Grabysz, A.: KMC 2: fast and resource-frugal k-mer counting. Bioinformatics (2015)
7. National Human Genome Research Institute (NIH): The Cost of Sequencing a Human Genome. (Available from: `https://www.genome.gov/sequencingcosts/`) (2016)
8. Tanenbaum, A.S., Van Steen, M.: Distributed systems. Prentice-Hall (2007)
9. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM 51, 107–113 (2008)
10. Apache Software Foundation: Hadoop. (Available from: `http://hadoop.apache.org/`) (2016)
11. Apache Software Foundation: Spark. (Available from: `http://spark.apache.org/`) (2016)
12. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. vol. 10, pp. 1–7 (2010)
13. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al.: Apache Hadoop YARN: yet another resource negotiator. In: Proceedings of the 4th annual Symposium on Cloud Computing. pp. 1–16. ACM (2013)
14. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop distributed file system. In: IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010. pp. 1–10. IEEE Computer Society, Washington, DC, USA (2010)
15. McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., et al.: The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Research 20, 1297–1303 (2010)
16. Nordberg, H., Bhatia, K., Wang, K., Wang, Z.: BioPig: a Hadoop-based analytic toolkit for large-scale sequence data. Bioinformatics 29, 3014–3019 (2013)
17. Matsunaga, A., Tsugawa, M., Fortes, J.: CloudBLAST: Combining MapReduce and virtualization on distributed resources for bioinformatics applications. In: IEEE Fourth International Conference on eScience, 2008. eScience'08. pp. 222–229. IEEE (2008)

18. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic Local Alignment Search Tool. Journal of Molecular Biology 215, 403–410 (1990)
19. Piotto, S., Di Biasi, L., Concilio, S., Castiglione, A., Cattaneo, G.: GRIMD: distributed computing for chemists and biologists. Bioinformation 10, 43–47 (2014)
20. Lopez, D.H., Fiol-deRoque, M.A., Noguera-Salvà, M.A., Terés, S., Campana, F., Piotto, S., Castro, J.A., Mohaibes, R.J., Escribá, P.V., Busquets, X.: 2-Hydroxy arachidonic acid: a new non-steroidal anti-inflammatory drug. PloS one 8, 1–10 (2013)
21. Piotto, S., Concilio, S., Bianchino, E., Iannelli, P., López, D.J., Terés, S., Ibarguren, M., Barceló-Coblijn, G., Martin, M.L., Guardiola-Serrano, F., Alonso-Sande, M., Funari, S.S., Busquets, X., Escribá, P.V.: Differential effect of 2-hydroxyoleic acid enantiomers on protein (sphingomyelin synthase) and lipid (membrane) targets. Biochimica et Biophysica Acta (BBA)-Biomembranes 1838, 1628–1637 (2014)
22. Piotto, S., Trapani, A., Bianchino, E., Ibarguren, M., López, D.J., Busquets, X., Concilio, S.: The effect of hydroxylated fatty acid-containing phospholipids in the remodeling of lipid membranes. Biochimica et Biophysica Acta (BBA)-Biomembranes 1838, 1509–1517 (2014)
23. Nguyen, T., Shi, W., Ruden, D.: CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping. BMC Research Notes 4, 171 (2011)
24. Cattaneo, G., Italiano, G.F.: Algorithm engineering. ACM Computing Surveys (CSUR) 31, 582–585 (1999)
25. Demetrescu, C., Finocchi, I., Italiano, G.F.: Algorithm engineering. Bulletin of the EATCS 79, 48–63 (2003)
26. Niemenmaa, M., Kallio, A., Schumacher, A., Klemelä, P., Korpelainen, E., Heljanko, K.: Hadoop-BAM: directly manipulating next generation sequencing data in the Cloud. Bioinformatics 28, 876–877 (2012)
27. Schönherr, S., Forer, L., Weißensteiner, H., Kronenberg, F., Specht, G., Kloss-Brandstätter, A.: Cloudgene: A graphical execution platform for MapReduce programs on private and public Clouds. BMC Bioinformatics 13, 1–9 (2012)
28. Ferraro Petrillo, U., Roscigno, G., Cattaneo, G., Giancarlo, R.: FASTdoop: A versatile and efficient library for the input of FASTA and FASTQ files for MapReduce Hadoop bioinformatics applications. Bioinformatics (2017), `https://dx.doi.org/10.1093/bioinformatics/btx010`
29. Schumacher, A., Pireddu, L., Niemenmaa, M., Kallio, A., Korpelainen, E., Zanetti, G., Heljanko, K.: SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop. Bioinformatics 30, 119–120 (2014)
30. Wiewiórka, M.S., Messina, A., Pacholewska, A., Maffioletti, S., Gawrysiak, P., Okoniewski, M.J.: SparkSeq: fast, scalable, Cloud-ready tool for the interactive genomic data analysis with nucleotide precision. Bioinformatics 30, 2652–2653 (2014)
31. Huang, H., Tata, S., Prill, R.J.: BlueSNP: R package for highly scalable genome-wide association studies using Hadoop clusters. Bioinformatics 29, 135–136 (2013)
32. Langmead, B., Schatz, M.C., Lin, J., Pop, M., Salzberg, S.L.: Searching for SNPs with Cloud computing. Genome Biology 10, 1–10 (2009)
33. Jourdren, L., Bernard, M., Dillies, M.A., Le Crom, S.: Eoulsan: a Cloud computing-based framework facilitating high throughput sequencing analyses. Bioinformatics 28, 1542–1543 (2012)
34. Hong, D., Rhie, A., Park, S.S., Lee, J., Ju, Y.S., Kim, S., Yu, S.B., Bleazard, T., Park, H.S., Rhee, H., et al.: FX: an RNA-Seq analysis tool on the Cloud. Bioinformatics 28, 721–723 (2012)
35. Langmead, B., Hansen, K.D., Leek, J.T., et al.: Cloud-scale RNA-sequencing differential expression analysis with Myrna. Genome Biology 11, 1–11 (2010)

36. Zhang, L., Gu, S., Liu, Y., Wang, B., Azuaje, F.: Gene set analysis in the Cloud. Bioinformatics 28, 294–295 (2012)
37. Almeida, J.S., Grüneberg, A., Maass, W., Vinga, S.: Fractal MapReduce decomposition of sequence alignment. Algorithms for Molecular Biology 7, 1–12 (2012)
38. Cattaneo, G., Ferraro Petrillo, U., Giancarlo, R., Roscigno, G.: An effective extension of the applicability of alignment-free biological sequence comparison algorithms with Hadoop. The Journal of Supercomputing pp. 1–17 (2016), `http://dx.doi.org/10.1007/s11227-016-1835-3`
39. Hill, C.M., Albach, C.H., Angel, S.G., Pop, M.: K-mulus: Strategies for BLAST in the Cloud. In: International Conference on Parallel Processing and Applied Mathematics. pp. 237–246. Springer (2013)
40. Colosimo, M.E., Peterson, M.W., Mardis, S., Hirschman, L.: Nephele: genotyping via complete composition vectors and MapReduce. Source Code for Biology and Medicine 6, 1–10 (2011)
41. Drew, J., Hahsler, M.: Strand: fast sequence comparison using MapReduce and locality sensitive hashing. In: Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics. pp. 506–513. ACM (2014)
42. Chang, Y.J., Chen, C.C., Chen, C.L., Ho, J.M.: A de novo next generation genomic sequence assembler based on string graph and MapReduce Cloud computing framework. BMC Genomics 13, 1–17 (2012)
43. Schatz, M.C., Sommer, D., Kelley, D., Pop, M.: De novo assembly of large genomes using Cloud computing. In: Proceedings of the Cold Spring Harbor Biology of Genomes Conference (2010)
44. Schatz, M.C.: BlastReduce: high performance short read mapping with MapReduce. University of Maryland, (Available from: `http://cgis.cs.umd.edu/Grad/scholarlypapers/papers/MichaelSchatz.pdf`) (2008)
45. Schatz, M.C.: CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics 25, 1363–1369 (2009)
46. Pireddu, L., Leo, S., Zanetti, G.: SEAL: a distributed short read mapping and duplicate removal tool. Bioinformatics 27, 2159–2160 (2011)
47. Zhao, G., Ling, C., Sun, D.: SparkSW: scalable distributed computing system for large-scale biological sequence alignment. In: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2015. pp. 845–852. IEEE (2015)
48. Leo, S., Santoni, F., Zanetti, G.: Biodoop: bioinformatics on Hadoop. In: International Conference on Parallel Processing Workshops, 2009 (ICPPW'09). pp. 415–422. IEEE (2009)
49. Radenski, A., Ehwerhemuepha, L.: Speeding-up codon analysis on the Cloud with local MapReduce aggregation. Information Sciences 263, 175–185 (2014)
50. Rasheed, Z., Rangwala, H.: A Map-Reduce framework for clustering metagenomes. In: IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013. pp. 549–558. IEEE (2013)
51. Matthews, S.J., Williams, T.L.: MrsRF: an efficient MapReduce algorithm for analyzing large collections of evolutionary trees. BMC Bioinformatics 11, 1–9 (2010)
52. Feng, X., Grossman, R., Stein, L.: PeakRanger: A Cloud-enabled peak caller for ChIP-seq data. BMC Bioinformatics 12, 1–11 (2011)