SparkBLAST: Introduction

Author : Muniba Faiza

Categories : Algorithms, Cloud Computing, Sequence Analysis

Date : July 13, 2017

The basic local alignment search tool (BLAST) [1,2] is known for its speed and results, which is also a primary step in sequence analysis. The ever increasing demand of processing huge amount of genomic data has led to the development of new scalable and highly efficient computational tools/algorithms. For example, MapReduce is the most widely accepted framework which supports design patterns representing general reusable solutions to some problems including biological assembly [3] and is highly efficient to handle large datasets running over hundreds to thousands of processing nodes [4]. But the implementation frameworks of MapReduce (such as Hadoop) limits its capability to process smaller data.

Cloud computing has emerged as a powerful tool to process these dynamic tasks over the last decade. Recently, a framework called Apache Spark emerged as a promising framework to implement highly scalable parallel applications [5,6]. A new parallelization of BLAST is developed by de Castro et al., 2017 employs cloud computing and Apache Spark as the coordination framework [7]. The SparkBLAST reduces the number of local input/output operations resulting in highly efficient and superior performance [7].

Working:

It requires two input files: **a)** a target database consisting of bacterial genomic sequences, and **b)** a query file consisting of a set of query genome sequences to be compared with the sequences of the target's database. The basic concept of the working of SparkBLAST is that as soon as it takes the input, it replicates the entire input database on every computing node, then it split the query file into fragments which are later evenly distributed to every node, thus, rendering each node with the local deployment of the BLAST application, a copy of the target database, and a fragment of the query sequences [7]. After that, the whole computation is partitioned into tasks by the (Spark's scheduler), which is assigned to the computing nodes depending on the data locality [8]. The replicated target database and the fragment of the query file are loaded into the memory for the execution of each task. The SparkBLAST then install the NCBI BLAST2 locally at each node with the help of Spark Pipe to execute multiple parallels and distributed tasks in the cluster. SparkBLAST uses YARN [9] as the resource manager to execute the tasks as it can be uniformly used by the Spark and Hadoop.

The processing of SparkBLAST is divided into three categories [7]:

1. Pre-processing

In this first stage, the query file is evenly partitioned into splits which are then distributed among the computing nodes.

2. Main-processing

After the data transferred to each processing node, the tasks are then scheduled on each computing node to be executed according to the data locality. At this stage, the NCBI BLAST2 has been installed locally at each node. As a computing core finishes a task, it starts working in a loop for all the tasks assigned to it and keeps doing until the available core executes all tasks of the assigned job.

3. Post-processing

During the second stage of processing, each individual task produces small output files, which are merged into a single output file at this stage.

Several other tools combine cloud and Hadoop technologies such as CloudBLAST [10], BioDoop [11], and Crossbow [12]. The SparkBLAST was found to outperform the CloudBLAST which is also a cloud platform [7,10].

For further details about SparkBLAST, click here.

References

1. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, *215*(3), 403-410.
2. Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., & Madden, T. L. (2009). BLAST+: architecture and applications. *BMC bioinformatics*, *10*(1), 421.

3. O'Driscoll, A., Daugelaite, J., & Sleator, R. D. (2013). 'Big data', Hadoop and cloud computing in genomics. *Journal of biomedical informatics*, *46*(5), 774-781.
4. Senger, H., Gil?Costa, V., Arantes, L., Marcondes, C. A., Marín, M., Sato, L. M., & Silva, F. A. (2016). BSP cost and scalability analysis for MapReduce operations. *Concurrency and Computation: Practice and Experience*, *28*(8), 2503-2527.
5. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *HotCloud*, *10*(10-10), 95.
6. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., … & Stoica, I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (pp. 2-2). USENIX Association.
7. de Castro, M. R., dos Santos Tostes, C., Dávila, A. M., Senger, H., & da Silva, F. A. (2017). SparkBLAST: scalable BLAST processing using in-memory operations. *BMC Bioinformatics*, *18*(1), 318.
8. Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010, April). Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European conference on Computer systems* (pp. 265-278). ACM.
9. Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., … & Saha, B. (2013, October). Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing* (p. 5). ACM.
10. Matsunaga, A., Tsugawa, M., & Fortes, J. (2008, December). Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on* (pp. 222-229). IEEE.
11. Leo, S., Santoni, F., & Zanetti, G. (2009, September). Biodoop: bioinformatics on hadoop. In *Parallel Processing Workshops, 2009. ICPPW'09. International Conference on* (pp. 415-422). IEEE.
12. Langmead, B., Hansen, K. D., & Leek, J. T. (2010). Cloud-scale RNA-sequencing differential expression analysis with Myrna. *Genome biology*, *11*(8), R83.

Sharing is caring. Spread the love!