Protein Cell 2012, 3(2): 148–152
DOI 10.1007/s13238-012-2015-8

Protein & Cell

RESEARCH ARTICLE

CloudLCA: finding the lowest common ancestor in metagenome analysis using cloud computing

Guoguang Zhao[1,4*], Dechao Bu[1,4*], Changning Liu[1], Jing Li[1], Jian Yang[3], Zhiyong Liu[1], Yi Zhao[1] ✉, Runsheng Chen[1,2] ✉

[1] Bioinformatics Research Group, Key Laboratory of Intelligent Information Processing, Advanced Computing Research Laboratory, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[2] Bioinformatics Laboratory and National Laboratory of Biomacromolecules, Institute of Biophysics, Chinese Academy of Sciences, Beijing 100101, China
[3] State Key Laboratory for Molecular Virology and Genetic Engineering, National Institute for Viral Disease Control and Prevention, Chinese Center for Disease Control and Prevention, Beijing 100176, China
[4] Graduate School of the Chinese Academy of Sciences, Beijing 100190, China
✉ Correspondence: chenrs@sun5.ibp.ac.cn (R. Chen), biozy@ict.ac.cn (Y. Zhao)
Received December 15, 2011 Accepted January 15, 2012

ABSTRACT

Estimating taxonomic content constitutes a key problem in metagenomic sequencing data analysis. However, extracting such content from high-throughput data of next-generation sequencing is very time-consuming with the currently available software. Here, we present CloudLCA, a parallel LCA algorithm that significantly improves the efficiency of determining taxonomic composition in metagenomic data analysis. Results show that CloudLCA (1) has a running time nearly linear with the increase of dataset magnitude, (2) displays linear speedup as the number of processors grows, especially for large datasets, and (3) reaches a speed of nearly 215 million reads each minute on a cluster with ten thin nodes. In comparison with MEGAN, a well-known metagenome analyzer, the speed of CloudLCA is up to 5 more times faster, and its peak memory usage is approximately 18.5% that of MEGAN, running on a fat node. CloudLCA can be run on one multiprocessor node or a cluster. It is expected to be part of MEGAN to accelerate analyzing reads, with the same output generated as MEGAN, which can be import into MEGAN in a direct way to finish the following analysis. Moreover, CloudLCA is a universal solution for finding the lowest common ancestor, and it can be applied in other fields requiring an LCA algorithm.

KEYWORDS CloudLCA, metagenome analysis, cloud computing

INTRODUCTION

Taxonomic content can reflect the conservation level of reads. Estimating such content was a key problem in the metagenomics analysis. This problem was first efficiently solved by Huson and Auch by introducing a Lowest Common Ancestor (LCA)-based algorithm (Huson et al., 2007). Since then, the LCA-based algorithm has been widely employed in metagenome analysis software programs designed to obtain taxonomic composition, such as MEGAN (Huson et al., 2007; Huson et al., 2009) and Galaxy (Blankenberg et al., 2007; Blankenberg et al., 2010). However, next-generation sequencing machines now produce an enormous amount of datasets, making effective analysis computationally challenging (Metzker, 2010; Qin et al., 2010). Recent studies propose cloud computing as a solution in a wide variety of biological analyses, such as read mapping (Schatz, 2009), multiple sequence alignment (Sudha Sadasivam and Baktavatchalam, 2010) and differential expression analysis (Langmead et al., 2010). Such biological analyses have gained superior performance through cloud computing, which,

*These authors contributed equally to the work.

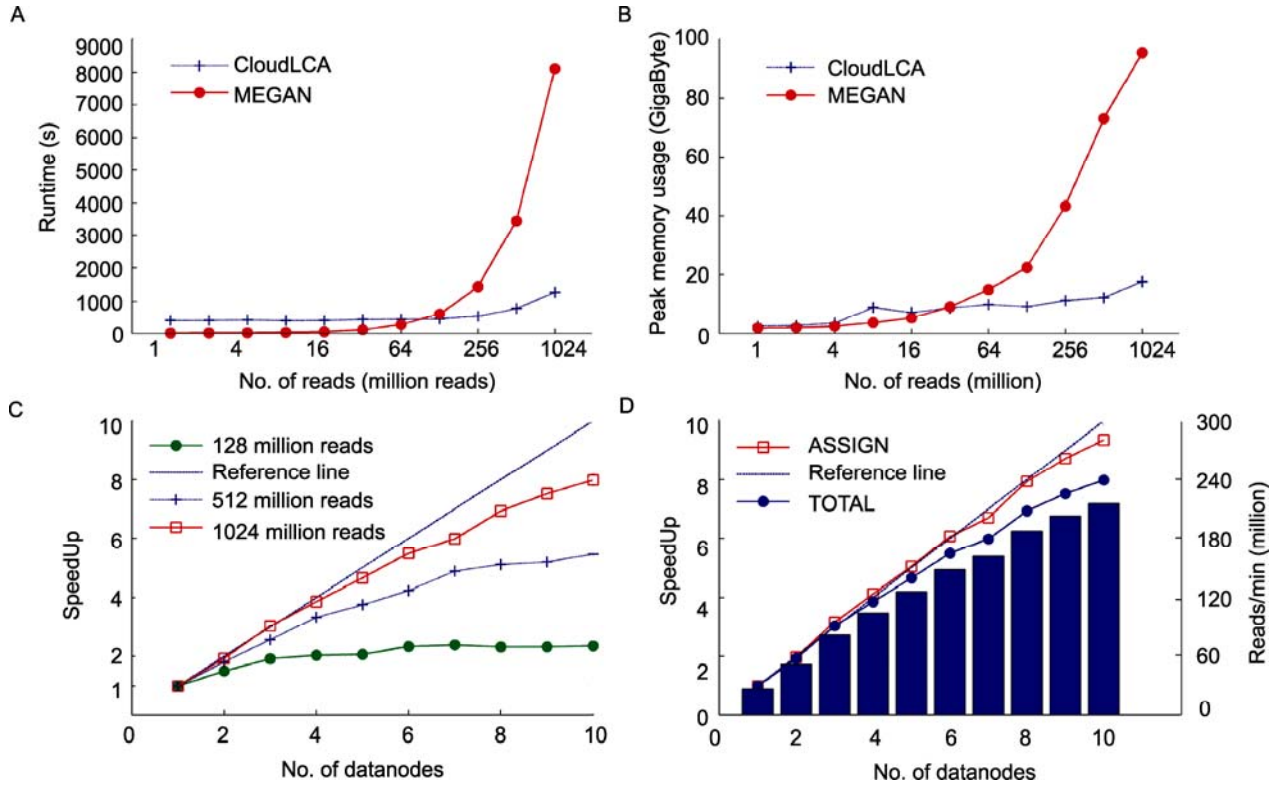

Figure 1. Scalability of CloudLCA (DDSs: Different Datasets; MPVs: Memory Peak Values). (A) Runtime of Megan and CloudLCA for DDSs. (B) PMU of Megan and CloudLCA for DDSs. (C) SpeedUp Over Different Datasets. And (D) Scalability of CloudLCA.

in turn, has benefited from using the open-source Hadoop (http://hadoop.apache.org) implementations of the MapReduce programming model (Lämmel, 2007), which is used to process large-scale datasets in a massively parallel manner. In this paper, we adopted an enhanced LCA-based algorithm and utilized the MapReduce model to develop one integrated strategy, termed CloudLCA. CloudLCA scales almost linearly to process the largest testing datasets on increasing thin nodes. In addition, CloudLCA can run up to 5 more times faster than MEGAN, while its peak memory usage is approximately 18.5% over that of MEGAN, both running on the fat node. Despite its speed, CloudLCA still exhibits high computational power in handling the larger testing datasets, and it is therefore an efficient solution for gaining the taxonomic composition for such large-scale datasets. Moreover, since CloudLCA is a universal solution for finding the lowest common ancestor, it is expected that CloudLCA can be applied in other fields for problems requiring an LCA algorithm, and it can be deployed on one multiprocessor node or a cluster.

RESULTS

In order to evaluate the performance of CloudLCA, we compared it with MEGAN and observed its scalability. The hardware and software configurations of the fat and thin nodes are given in Table 1. Twenty-two datasets were used in the comparison, 11 of which were simple datasets, and the other 11 were complex datasets (see Simulation dataset section of DESIGN AND IMPLEMENTATION).

First, we ran both CloudLCA and MEGAN4 (Huson et al., 2011) (version 4.40.6) on the fat node to conduct an analysis of the simple and complex datasets, respectively. Multiprocessor programming was taken into account for both software programs. Results demonstrate that both of them can process 1024 million reads using the fat node, while CloudLCA is more than 5 times faster than MEGAN with a peak memory usage of approximately 18.5% over that of MEGAN (Fig. 1A and 1B). Currently, the maximum RAM consumed by MEGAN reaches 95.6 gigabytes, or 37.3% of

Table 1 Configurations of the thin nodes and fat nodes

Name	No. of CPU cores	CPU type	Memory	Operating system	Hadoop version
Thin node	8	Xeon E5430 2.66GHz	16 GB	SuSE linuxl 0	0.20.2
Fat node	64	Intel(R) Xeon(R) CPU X7550 2.00GHz	256 GB	CentOS 5.3	0.20.2

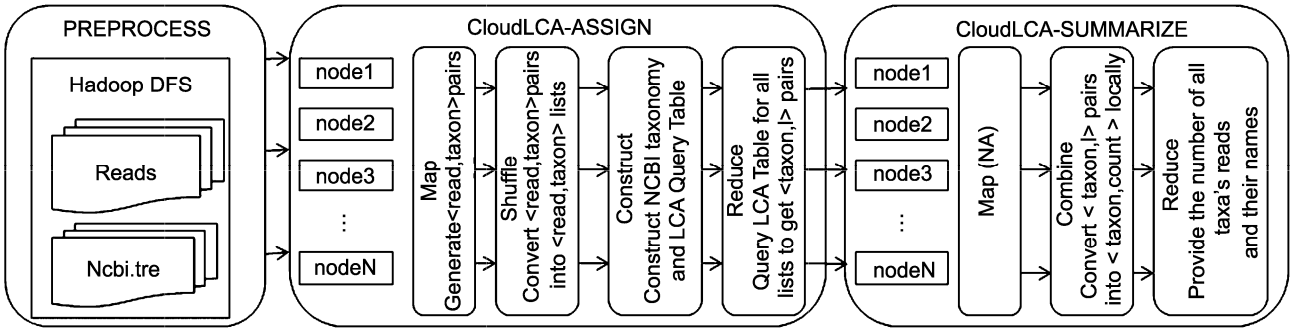

Figure 2. CloudLCA workflow. CloudLCA workflow is divided into three phases: PREPROCESS, ASSIGN and SUMMARIZE, including two MapReduce processes.

total RAM, while CloudLCA is about 17.7 gigabytes, or 6.9% of total RAM. Moreover, CloudLCA shows high scalability in handling the largest testing datasets. The results also show that the running times of both CloudLCA and MEGAN are identical, irrespective of the size of data or whether simple or complex datasets are involved.

Secondly, CloudLCA operated separately from one to ten (thin) worker nodes, with one new worker node added for each run. Then, three different sizes of datasets ("128 million," "512 million" and "1024 million") were used to evaluate CloudLCA. When the data size reached 1024 million reads, CloudLCA presented the fastest acceleration performance, close to linear speedup (Fig. 1C). This result implies that CloudLCA is capable of processing the largest datasets. The results also demonstrate that the ASSIGN phase holds perfect acceleration performance, obviously better than the TOTAL phase (TOTAL=ASSIGN+SUMMARIZE) (Fig. 1D). Meanwhile, we also demonstrate a correlation between the nearly linear increase in CloudLCA scalability and the number of worker nodes growing linearly for the same size of dataset (“1024 million”). The maximum data processing speed can reach 215 million reads per minute (Fig. 1D; histogram). At the same time, it is worth mentioning that peak memory usage of all worker nodes is between 4.8 gigabytes and 5.4 gigabytes, or 30% of total memory.

Here we provided an example (Yang et al., 2011) of biological problems solved using the software. In this paper, the Bowtie program was used to align each short sequence from an Illumina GA II sequencer to sequences in the nonredundant nucleotide (NT) database from GenBank. A series of in-house Shell scripts were then employed for formatting results in input datasets as Testing datasets. The results were then imported into CloudLCA software to assign each sequence to the LCA of the set of taxa that it hits. The command here:

$ CloudLCA-treefile ncbi.tre-inputfile example.csv example-summarize-result.out

The option-treefile specifies the NCBI taxonomic tree file; the option-inputfile specifies the input data as Testing dataset format; the example-summarize-result.out specifies the output file.

The example-summarize-result.out file was then imported into MEGAN4 software to do downstream analysis.

Besides, how to install and run CloudLCA was provided in the CloudLCA-software package.

DESIGN AND IMPLEMENTATION

MapReduce design

The LCA algorithm finds the lowest common ancestor for taxa with the same read assigned and determines the taxonomic content for large-scale datasets. In this paper, we want to demonstrate the effect integrating the RMQ (Range Minimum Query)-based LCA algorithm with the MapReduce programming model because (1) MapReduce has been successfully applied to the processing of large-scale datasets in a massively parallel manner and (2) the RMQ-based LCA algorithm is compatible with the MapReduce programming model. Essentially, by reducing the original LCA problem to an RMQ problem, a Sparse Table (ST) algorithm is introduced with complexity of <O(nlogn); O(1)>, where O(nlogn) is the running time complexity of constructing a sparse query table and O(1) is the running time complexity of querying. In our design, the query is to find the lowest common ancestor of any pair or group of nodes, but the precondition to constructing the sparse table in accordance with the RMQ-based LCA algorithm, as described above, involves the pre-construction of an LCA query table for pairs of nodes in the NCBI taxonomy tree. Satisfying the RMQ-based LCA problem in this manner allows us, in turn, to take advantage of the MapReduce programming model, thus creating a seamlessly integrated parallel processing system, guaranteeing the scalability of CloudLCA.

CloudLCA workflow is divided into three phases: PREPROCESS, ASSIGN and SUMMARIZE, including two MapReduce processes (Fig. 2).

PREPROCESS:CloudLCA copies Ncbi.tre (NCBI taxonomy tree file) and testing datasets to a Hadoop Distributed File System (HDFS), making them visible to the Hadoop cluster platform.

ASSIGN:CloudLCA constructs a query Sparse Table (ST) and finds the lowest common ancestor for taxa with the same read assigned. The detailed algorithm is demonstrated in Fig. 3A.

SUMMARIZE:CloudLCA provides the number or the names of reads for every related taxon. In "SUMMARIZE-Combine" and "SUMMARIZE-Reduce." Finally, CloudLCA separately outputs local statistical results and the taxonomic content of the test data. The detailed algorithm is demonstrated in Fig. 3B.

Simulation datasets

Input of CloudLCA is produced by BLAST or some other sequence comparison methods with the following format:

Read2345, 9606, 62
Read432, 13443, 61
Read33, 9606, 60
Read344, 9606, 61
Read565, 9606, 70
Read226, 9606, 60
……

Each row is a 3-tuple (readID, taxonID, score) separated by a comma, where readID is a unique name or ID for each read, taxonID is a taxon name or an ID of NCBI taxonomy tree, and score is from BLAST or some other sequence comparison methods for this alignment. Taking the 3-tuple "Read2345, 9606, 62" for example, the read Reads2345 matches the taxon 9606 with a bit score of 62. Reads have a many-to-many relationship with taxa in NCBI taxonomy tree. It is because that a read that has hits with sequences belongs to two or more taxa.

A

```
Setup(){
      ST=STConstruct("Ncbi.tre");
      // Construct a query Sparse Table (ST) using
      // the tree file "Ncbi.tre"
}
Map(){
      Print <readID, taxonID>
      // readID is a unique name or ID for each read
      // taxonID is a taxon name or an ID of NCBI taxonomy tree
 }
Reduce(){
      LCA=RMQQuery(ST, taxonIDs);
      //Query ST to find their lowest common ancestor (LCA)
      //for all taxonIDs.
      // LCA is also a taxonID of NCBI taxonomy tree
      Print <LCA, readID>;
}
```

B

```
Map(){
      Print <taxonID, readID>
}
Reduce(){
      Count=count(taxonID, readIDs);
      //To summarize the number or the names of reads
      //for every related taxon
      Print <taxonID, Count>;
}
```

Figure 3. Code segment showing the MapReduce implementation using Hadoop. (A) Code segment of ASSIGN process. (B) Code segment of SUMMARIZE process.

22 datasets have already been generated to simulate real datasets as to test the performance of CloudLCA and further illustrate the capability of CloudLCA. The algorithm to generate datasets is based on the following rules: (1) No duplicate 3-tuples. Duplicate tuples exist in the real-life metagenomic data. We discard the duplicate tuples because they only enlarge the data size, but not contribute to increasing CloudLCA's computational complexity. (2) One read corresponding to 100 taxa on average. According to statistics of real data, the number of the corresponding taxa of 70% of the reads is less than 100. (3) Generating two types of testing datasets: the simple dataset, the complex dataset.

Consequently, the datasets generated have the following characteristics: (1) No duplicate 3-tuples. (2) For any read in the datasets, the number of its corresponding taxa fluctuates around 100. According to statistics of all the testing datasets, the number of the corresponding taxa of approximately 96% of the reads is between 80–120. (3) The number of tuples generated covers four orders of magnitude, from million to billion. If length of each read is 90 bps long, then length of sequences will be from 90 million bps to 90 billion bps long, the equivalent of length of 90 microbial genomes (if there are one million base pairs for each microbe) or 30 human genomes.

Two algorithms written in PERL scripts to generate testing datasets are included in "CloudLCA" program packages. Each dataset is generated 5 times in order to avoid the occurrence of random situations when testing performances of CloudLCA and MEGAN.

A case study: real data analysis

To illustrate the use of CloudLCA, we apply CloudLCA to the analysis of a metatranscriptome of Nasopharyngeal aspirates (NPAs) with 16 samples and 105,891,205 (valid sequences in total from (Yang et al., 2011)). The Bowtie program was used to efficiently align these short sequences to reference sequences in the nonredundant nucleotide (NT) database from GenBank (downloaded in September 2010) with the parameter "-a --best" to allow the program to report all equally best matches with up to 4 mismatches. The alignment results were then imported into CloudLCA to assign each sequence to the LCA of the set of taxa that it hits. Result demonstrates that it takes CloudLCA less than 40 s to find LCA for the alignment of 105,891,205 valid sequences running on the fat node.

Availability and future directions

The implementation of CloudLCA, data generation scripts, sample datasets, and usage instructions are available online at http://www.ebiomed.org/CloudLCA-software.zip

ACKNOWLEDGEMENTS

We are grateful to Shiwei Sun (Bioinformatics Research Group, Key Laboratory of Intelligent Information Processing, Advanced Computing Research Laboratory, Institute of Computing Technology, Chinese Academy of Sciences, China) and Xiaopeng Zhu (Bioinformatics Laboratory, Institute of Biophysics, CAS, Beijing, China) for inspirational discussion in analysis of metagenomic data.

REFERENCES

Blankenberg, D., Von Kuster, G., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A., Taylor, J. (2010). Galaxy: a web-based genome analysis tool for experimentalists. Curr Protoc Mol Biol, Chapter 19, Unit 19.10 11–21.

Blankenberg, D., Taylor, J., Schenck, I., He, J., Zhang, Y., Ghent, M., Veeraraghavan, N., Albert, I., Miller, W., Makova, K.D., et al. (2007). A framework for collaborative analysis of ENCODE data: making large-scale analyses biologist-friendly. Genome Res 17, 960–964.

Huson, D.H., Auch, A.F., Qi, J., and Schuster, S.C. (2007). MEGAN analysis of metagenomic data. Genome Res 17, 377–386.

Huson, D.H., Mitra, S., Ruscheweyh, H.J., Weber, N., and Schuster, S.C. (2011). Integrative analysis of environmental sequences using MEGAN4. Genome Res 21, 1552–1560.

Huson, D.H., Richter, D.C., Mitra, S., Auch, A.F., and Schuster, S.C. (2009). Methods for comparative metagenomics. BMC Bioinformatics 10, S12.

Lämmel, R. (2007). Google's MapReduce programming model - Revisited. Sci Comput Program 68, 208–237.

Langmead, B., Hansen, K.D., and Leek, J.T. (2010). Cloud-scale RNA-sequencing differential expression analysis with Myrna. Genome Biol 11, R83.

Metzker, M.L. (2010). Sequencing technologies - the next generation. Nat Rev Genet 11, 31–46.

Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K.S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., et al., and the MetaHIT Consortium. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. Nature 464, 59–65.

Schatz, M.C. (2009). CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics 25, 1363–1369.

Sudha Sadasivam, G., and Baktavatchalam, G. (2010). A novel approach to multiple sequence alignment using hadoop data grids. Int J Bioinform Res Appl 6, 472–483.

Yang, J., Yang, F., Ren, L., Xiong, Z., Wu, Z., Dong, J., Sun, L., Zhang, T., Hu, Y., Du, J., et al. (2011). Unbiased parallel detection of viral pathogens in clinical samples by use of a metagenomic approach. J Clin Microbiol 49, 3463–3469.

