HTSeq-Hadoop: Extending HTSeq for Massively Parallel Sequencing Data Analysis using Hadoop

Alexey Siretskiy
Department of Information Technology
Uppsala University
SE-751 05 Uppsala
Email: alexey.siretskiy@it.uu.se

Ola Spjuth
Department of Pharmaceutical Biosciences
and Science for Life Laboratory
Uppsala University
SE-751 24 Uppsala
Email: ola.spjuth@farmbio.uu.se

***Abstract*—Hadoop is a convenient framework in e-Science enabling scalable distributed data analysis. In molecular biology, next-generation sequencing produces vast amounts of data and requires flexible frameworks for constructing analysis pipelines. We extend the popular HTSeq package into the Hadoop realm by introducing massively parallel versions of short read quality assessment as well as functionality to count genes mapped by the short reads. We use the Hadoop-streaming library which allows the components to run in both Hadoop and regular Linux systems and evaluate their performance in two different execution environments: A single node on a computational cluster and a Hadoop cluster in a private cloud. We compare the implementations with Apache Pig showing improved runtime performance of our developed methods. We also inject the components in the graphical platform Cloudgene to simplify user interaction.**

I. Introduction

The emergence of massively parallel sequencing (MPS), also referred to as next-generation sequencing, has made molecular biology data-intensive and dependent on high-performance computing (HPC), large-scale storage and a large flora of bioinformatics software applications [1], [2].

In MPS, generally, one fragments the DNA (DNA-seq experiment) or RNA (RNA-seq experiment) of the biological sample into short fragments of commonly between 300-400 nucleotides by sonication. The DNA (or its complementary – cDNA – for RNA-seq) fragments are cloned in parallel by the sequencing instrument, resulting in millions or billions of 'short reads' of 100-200 nucleotides long, and the sequence of each is determined.

Computational algorithms can be used thereafter to map the reads to a reference sequence and identify mutations (single nucleotide polymorphisms or SNPs) for DNA-seq or to calculate the gene expression levels for RNA-seq experiments. The analyses performed on short reads are generally computationally expensive and preferably carried out on high-performance computers [3]. The capacity of modern MPS-instruments has increased dramatically in recent years; for example, an Illumina high-throughput sequencing run produces approximately 1800 gigabases (Gbases) corresponding to 2 terabytes (TB) of raw data in 3 days [4].

MPS has revolutionized molecular biology and is widely used in, e.g., cancer research [5] and shows great potential for use in clinical applications such as for personalizing diagnostics and treatments [6]. However at present our ability to sequence DNA far exceeds our ability to analyze and apply the results to clinical care, which has spawned the expression 'The $1,000 genome, the $100,000 analysis' [7], and bioinformatics challenges are now widely recognized as a bottleneck in many applications of MPS [8].

Current HPC systems are mainly designed for fast calculations on small or moderate amounts of data; there is often InfiniBand fabric connecting powerful multicore nodes with tens or hundreds of GB of RAM and RAID-based network attached storage (NAS). Corresponding libraries for harnessing hundreds of cores are mainly message-passing interface (MPI)-based, and not suited for managing hundreds of GBs of data. This infrastructure did not arise by random, but instead was built up to fulfill the requirements in disciplines with a longer tradition in HPC such as physics, chemistry and astronomy. A common type of problem in physics is, e.g., to model a system with a moderate number of components (particles, blobs etc.) often yielding a squared number of interactions to calculate, and thousands of CPU-hours of calculations for a small input file.

MPS analysis contrasts to traditional HPC usage with the focus on digesting large scale data and while the majority of MPS data analysis currently is executed on Linux servers or clusters using batch systems, a recent alternative has emerged in the Hadoop framework [9]. While traditional HPC has focused on performance, the main argument for Hadoop is its scalability to operate on many thousands of nodes. With the MapReduce (MR) programming model [10] and Hadoop Distributed File System (HDFS), Hadoop excels in analyzing large data sets applying data partitioning and executing computations close to their data to avoid shuffling data between nodes [11].

In this report we describe `HTSeq-Hadoop` which extends the standard `HTSeq` package [12] with Hadoop implementations. `HTSeq` provides an Application Programming Interface (API) to manipulate raw and processed MPS data using the Python programming language. A limitation of `HTSeq` is that it is generally restricted to a single thread, though allowing to scale up to a whole multicore node in some cases.

As a starting point, we modified two widely used tools from

978-1-4799-4287-9/14 $31.00 © 2014 IEEE
DOI 10.1109/eScience.2014.27

HTSeq in RNA-seq analysis: htseq-count for counting how many reads are mapped to the genes and htseq-qa for quality assessment of raw or mapped reads. These were adapted to run in the Hadoop framework in order to significantly increase the scalability. The runtime performance of HTSeqCount under Hadoop was compared with the Pig Latin script on the Apache Pig platform [13]. The choice of Hadoop-streaming allowed us to reuse the code and involve the GNU-parallel [14] utility to run HTSeq-Hadoop in a usual Linux environment in multiple threads on the multicore workstations or on a cluster node.

Hadoop has been used for analyzing data from MPS in various projects and applications [15]. For analyzing gene expression data using the RNA-seq methodology, the Hadoop-enabled application Myrna [16] is available. The standard htseq-count prepares data to be further processed with the DESeq or EdgeR packages [17], which use a slightly different approach for calculating differential expression, providing an alternative. SeqPig [18] and BioPig [19] provide extensions to the Apache Pig language to support large sequence analysis tasks that run in parallel on the nodes of a Hadoop cluster. These lack ready-to-use scripts to mimic the gene-count functionality, which we complete in this work.

The rest of the paper is organized as follows. Section II presents the standard HTSeq package and Section III describes our implementation of HTSeq in Hadoop. Section IV presents the evaluation of the implementation and comparison with other similar tools. In Section V we discuss usability aspects. Section VI, followed by the Supplementary material concludes the paper.

II. HTSeq

The HTSeq package is a Python library with an API for dealing with sequencing data. The package contains a set of classes which are highly suitable for accessing MPS data, like reading and writing common sequence file formats including SAM, BAM, GFF and FASTQ, and has been used in experiments on gene expression analysis [17], [20]. The methods provide a convenient way to accumulate various statistics with a possibility to produce quality vector graphics. A drawback of the HTSeq package is that it does not natively support distributed calculations.

The HTSeq package provides the utility htseq-count to count how many reads map to each feature (such as a gene) for the given dataset. This very general kind of information is needed as a step for a gene expression analysis pipeline in RNA-seq experiments. Another utility is htseq-qa for quality assessment of short reads or for already mapped ones. Its output gives an insight into the quality of the raw or mapped reads, along with the complementary nucleotide proportions that must satisfy certain criteria to ensure a properly executed experiment. Complete documentation for HTSeq can be found on the Web [21].

III. HTSeq-Hadoop

We developed MapReduce (MR) implementations of the HTSeq applications htseq-count and htseq-qa since they have a great potential for a parallelization and are needed in many types of bioinformatics analysis. We used the Hadoop-streaming library as it allows for writing mappers and reducers in any programming/scripting language. The library also provides a possibility to use the STDIN and STDOUT to be engaged by Hadoop for massively parallel calculations on a Hadoop cluster. The HTSeq-Hadoop package presently contains two Hadoop-MR programs: HTSeqCount and HTSeqQA which are mimicking the functionality of htseq-count and htseq-qa correspondingly. An example output of the HTSeqQA function can be seen in Fig. 1. Each of the Hadoop programs consists of a mapper.py and reducer.py written in the Python programming language, receiving input via STDIN and writing to STDOUT. A brief sketch of the streaming library usage is given in Listing 1.

The mappers and reducers of HTSeq-Hadoop are based heavily on the refactored code of the HTSeq package. The HTSeqCount's mapper is the Python wrapper for the htseq-count. The HTSeqCount's reducer collects the partial gene counts from all the mappers in order to produce the final score. In order to run htseq-qa with Hadoop, we adopted and modified the original code. The mapper reads the SAM file and creates (key, value) pairs, where each key is the label assigned to a row of the quality matrix and value are the corresponding row entries. The quality matrix is a table, containing counts of called bases for each position in the read. I.e., for 100bp reads the matrix will have 100 rows and 5 columns with counts for "A", "T", "G", "C" for corresponding nucleotides and "N" for unknowns. The reducer assembles the matrix, normalizes the data, and produces the SVG graphic output, as shown on the Fig. 1.

Listing 1: Schematic listing of the Hadoop-streaming library usage

```
#invoking Hadoop with the streaming library
user@ubuntu$ hadoop jar /pathTo/streaming.jar \
#local files on the Hadoop client
-file /localPath/mapper.py
-mapper /localPath/mapper.py \
-file /localPath/reducer.py
-reducer /localPath/reducer.py \
#input and output folders on the HDFS
-input /user/hduser/input \
-output /user/hduser/output
```

Listing 2: Running HTSeqCount with GNU-parallel

```
f=file.sam.bz2
pbzip2 -dc $f | parallel -k --pipe \
grep -E -v -i 'mitochondria|chloroplast' |\
parallel --no-notice --pipe --files --block 100M \
./HTSeqCount_mapper.py -gff ../ -t gene -i ID |\
parallel -Xj1 --no-notice sort -m {} ';' rm {} |\
./HTSeqCount_reducer.py >./$f.sorted
2>>logCountParallel
```

An advantage of using the Hadoop streaming library is that our implementation can be run on the more commonly available Linux workstations without Hadoop installed. For example, one can use `GNU-parallel` with `HTSeqCount` to utilize all the cores on the multi-node machine as shown in Listing 2. A more detailed description of `HTSeq-Hadoop` is given in the online documentation [22].

IV. Performance evaluation

A. Test data

In order to test the `HTSeq-Hadoop` implementation we used a publicly available dataset for the model plant *A. thaliana* (Dataset I) and a synthetic dataset created with the Samtools package [23] (Dataset II), see Supplementary Material for full details. The size of Dataset I is 30 Gbases (billions of nucleotides) and Dataset II 100 Gbases; for comparison, the human genome contains about 3 Gbases. The physical sizes of the datasets are 21 and 38 GB, correspondingly, in FASTQ format compressed with `bzip2`. Bowtie [24] was used for mapping the datasets, see Supplementary Material for details.

B. Computational resources

Our Hadoop test platform was deployed on a private cloud using the OpenNebula [25] cloud management system. Each of 54 physical nodes was equipped with two 4-core Intel Xeon 5420 (2.50 GHz; 12 MB L2 cache), 16 GB RAM, one 1 TB SATA disk and Gigabit Ethernet. We used the Cloudera Distribution Hadoop (CDH), Hadoop 2.0.0-cdh4.6.0 [26], installed using Puppet [27] as a configuration manager tool.

We also used an HPC cluster where each node was equipped with two 8-core Intel Xeon E5-2660, (2.2 GHz, 2 MB L2 cache, 20 MB L3 cache), 128 GB of RAM, Infiniband node-to-node network connection and 10 Gbit/s uplink.

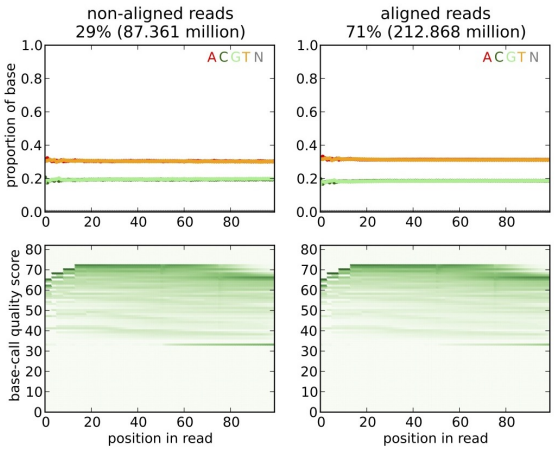


Fig. 1: Graphical output of the `HTSeqQA`: quality assessment results for dataset I, mapped to the reference genome. The left column is made from the non-mapped, the right column from the mapped reads, with the corresponding number of reads. As we can see the mapped data satisfies the complementarity rule: the proportion of "A" is equal to that of "T", and the same is true for "G" and "C".

C. Single node performance

We used `GNU-parallel` software to run the `HTSeqCount` and `HTSeqQA` utilities using multiple threads. `GNU-parallel` is a free Linux tool with Command Line Interface (CLI) for splitting the data for calculation-intensive tasks into chunks and executing these in parallel on a multicore computing node [14]. A linear scaling of the multithreading parallelization for the `HTSeqQA` can be observed in Table I. The performance for the `HTSeqCount` implementation is shown in Table II.

TABLE I: Timings for the `HTSeqQA` executed on a single node with the `GNU-parallel` for different number of lines from the the dataset I, in minutes.

number of lines	10×10^6	50×10^6	100×10^6	292,985,136 (whole dataset)
timing	0.66	2.47	4.79	13.94
	0.63	2.44	4.77	13.87
	0.61	2.45	4.76	13.93
average and stdev.	**0.63±0.02**	**2.45±0.01**	**4.77±0.01**	**13.91±0.04**

TABLE II: Timings for the `HTSeqCount` executed on a single core and a node with the `GNU-parallel` for the dataset I, in minutes.

dataset	`HTSeqCount` on a single core	`HTSeqCount`, 16 cores with `GNU-parallel`
	543.05	30.75
I	535.38	30.54
	557.37	30.36
average and stdev.	**545.27±11.16**	**30.51±0.18**

The resulting gene counts from the `HTSeqCount` using `GNU-parallel` and from the original single-threaded application differ in only 78 places for Dataset I (see Supplementary Material) with no practical impact on the gene expression analysis results. The difference arises due to the fact that `GNU-parallel` uses the `dd` Unix utility to split the data into chunks, which cuts based on the size in bytes, raising conflicts with string-oriented formats like SAM.

D. Hadoop cluster performance

The major decrease in execution time due to massively parallel execution can be observed for the code being run on the Hadoop cluster. Table III shows the `HTSeqQA` timings for Datasets I and II executed on a single node and on a Hadoop cluster. Similarly, Table IV provides the timings for `HTSeqCount`. As for execution with `GNU-parallel`, `HTSeqCount` gives perfect consistency with the original single-threaded version. Note that since the block size used for Hadoop (128 MB of `bzip`'ed data) is greater than the 100 MB of unarchived data used for `GNU-parallel`, there are

TABLE III: Timings for HTSeqQA executed on a single node with GNU-parallel and on the Hadoop cluster for datasets I and II, in minutes.

dataset	HTSeqQA, (single node with 16 cores, GNU-parallel)	HTSeqQA (Hadoop cluster, 54 cores)
I	13.94	6.30
	13.87	5.88
	13.93	5.82
average and stdev.	**13.91±0.04**	**6.00±0.26**
II	40.38	14.53
	40.43	14.50
	40.00	14.48
average and stdev.	**40.27±0.24**	**14.51±0.01**

TABLE IV: Timings for HTSeqCount executed on a single node with GNU-parallel and on the Hadoop cluster and Apache Pig (for the *intersection-strict* htseq-count mode) for the datasets I, half of II, and II, in minutes.

mapped dataset	HTSeqCount (single node with 16 cores, GNU-parallel)	HTSeqCount (Hadoop cluster, 54 cores)	Apache Pig, 54 cores
I	30.75	8.47	12.87
	30.54	8.48	12.95
	30.36	8.48	12.95
average and stdev.	**30.51±0.18**	**8.48±0.01**	**12.92±0.05**
half of dataset II	65.87	12.83	19.23
	65.60	12.73	19.17
	65.28	12.80	19.25
average and stdev.	**65.58±0.29**	**12.79±0.05**	**19.22±0.04**
II	102.31	22.45	38.20
	165.86	22.45	38.12
	178.14	22.50	38.35
average and stdev.	**148.77±40.70**	**22.47±0.03**	**38.22± 0.12**

14 different counts compared to original version for Dataset I, see Supplementary Material.

E. Comparison with Apache Pig

The Apache Pig platform is for analyzing large data sets on Hadoop in a high-level language similar to that of SQL for relational database management systems. Apache Pig translates the queries into a sequence of MR jobs which are sequentially executed on the underlying Hadoop platform. A key point with Apache Pig is that no Java knowledge is needed and the native Pig syntax is easy to learn. However, there are some specifics one has to take into account. Apache Pig operates in a manner so that each query involves data stored on all nodes. This gives some definite advantages for, e.g., easy global sorting. The disadvantages are related to the increasing number of MR-jobs, i.e., not all data operations can be efficiently formulated in a MR-manner.

We implemented a Pig Latin script mimicking the htseq-count functionality for the *intersection-strict* mode, which was easiest to implement among the *union, intersection-strict, intersection-union*. We then evaluated its performance on a Hadoop cluster and compared with the HTSeqCount implementation (see Table IV for the results). We observe that the Apache Pig script in its current form is outperformed by HTSeqCount, while still performing much better than HTSeqCount run on a single node.

Other Pig-based packages, like BioPig and SeqPig provide high level functions aimed to ease the API to the MPS data. There were, however, no ready to use functions suitable for our purposes in BioPig, and the current release of SeqPig gives version incompatibility issues on our Hadoop cluster.

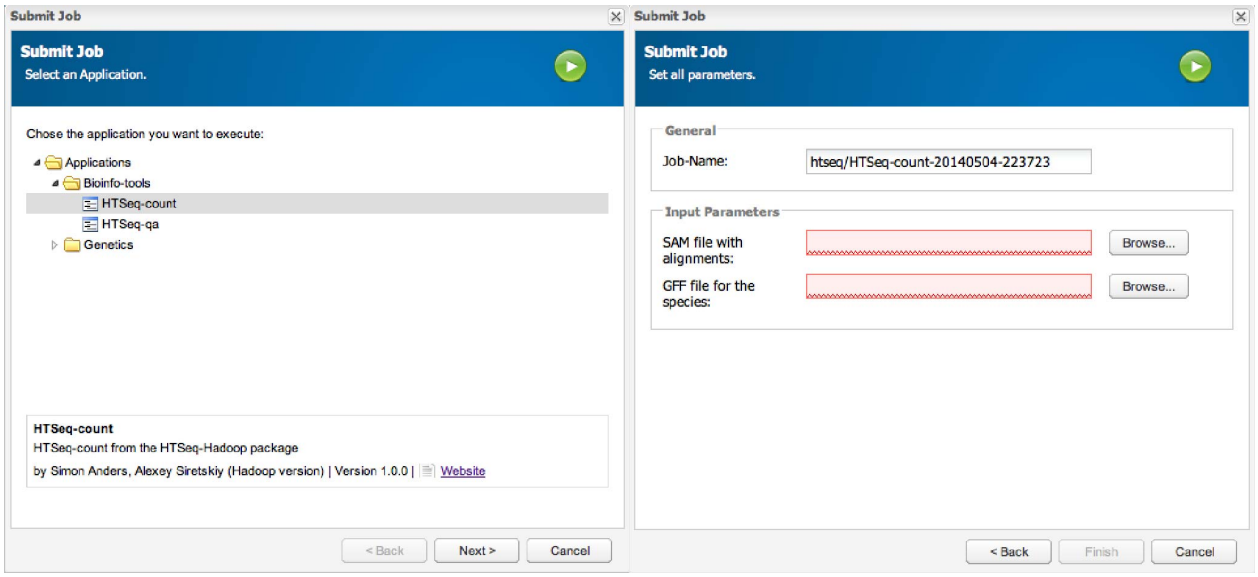
V. Usability aspects

We implemented HTSeq-Hadoop in the graphical platform Cloudgene [28], in order to provide a smoother user experience. Cloudgene is a lightweight and well-supported free software package to orchestrate and execute Hadoop pipelines and provides a simple HDFS browser. The results of the Hadoop calculations can be downloaded to the local machine or observed directly in the Internet browser. Having an easy, graphical way to invoke HTSeq-Hadoop should greatly increase the uptake in the biological research community where competence in Linux scripting is not ubiquitous. The interface for executing HTSeq-Hadoop tasks with Cloudgene is shown in Fig. 2.

VI. Discussion and conclusion

In this correspondence we present the software package HTSeq-Hadoop. We show how two HTSeq utilities htseq-count and htseq-qa can be modified and wrapped in order to operate in the Hadoop framework and benefit from massively parallel execution. These utilities are general purpose tools, widely used in RNA-seq experiments analysis.

For htseq-count we investigate different scenarios to use Hadoop: to wrap the already existing code and engage



(a) job selection

(b) specifing the job input files

Fig. 2: A graphical user interface for the `HTSeq-Hadoop` package with the Cloudgene platform.

Hadoop-streaming library or to write a mimicking program from scratch using the Apache Pig Latin language. These implementations are independent of each other and provide some understanding of which computational strategy is favorable for the given task – to count genes mapped by the short nucleotide sequences generated in MPS experiments.

The first scenario, presented in the `HTSeq-Hadoop` package, appears to be more efficient (less time consuming). The key role is a trivial parallelization of the gene-counting task; it is possible to split the dataset and count features independently for each chunk of the data. However the Pig Latin approach clearly demonstrates the power of the scripting with Pig – the code for the program is less than 50 lines long, showing almost linear scaling of the calculation time with the data size.

The source code for the `HTSeq-Hadoop` package and the Apache Pig implementation of gene counting along with the documentation are publicly available on GitHub [29], [30].

Acknowledgment

The computations were performed on resources provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (SNIC-UPPMAX) under project p2013023. This work was supported by the Swedish strategic research programme eSSENCE and COST Action BM1006 SeqAhead. We thank Wesley Schaal for proofreading.

References

[1] M. L. Metzker, "Sequencing technologies – the next generation," Nat Rev Genet, vol. 11, no. 1, pp. 31–46, 2010.

[2] M. Baker, "Next-generation sequencing: adjusting to data overload," Nat Meth, vol. 7, no. 7, pp. 495–499, 2010.

[3] S. Lampa, M. Dahlö, P. Olason, J. Hagberg, and O. Spjuth, "Lessons learned from implementing a national infrastructure in Sweden for storage and analysis of next-generation sequencing data," GigaScience, vol. 2, no. 1, p. 9, 2013.

[4] "Hiseq Comparison." [Online]. Available: http://www.illumina.com/systems/sequencing.ilmn

[5] G. M. Frampton, A. Fichtenholtz, G. A. Otto, K. Wang, S. R. Downing, J. He, M. Schnall-Levin, J. White, E. M. Sanford, P. An, J. Sun, F. Juhn, K. Brennan, K. Iwanik, A. Maillet, J. Buell, E. White, M. Zhao, S. Balasubramanian, S. Terzic, T. Richards, V. Banning, L. Garcia, K. Mahoney, Z. Zwirko, A. Donahue, H. Beltran, J. M. Mosquera, M. A. Rubin, S. Dogan, C. V. Hedvat, M. F. Berger, L. Pusztai, M. Lechner, C. Boshoff, M. Jarosz, C. Vietz, A. Parker, V. A. Miller, J. S. Ross, J. Curran, M. T. Cronin, P. J. Stephens, D. Lipson, and R. Yelensky, "Development and validation of a clinical cancer genomic profiling test based on massively parallel dna sequencing," Nat Biotechnol, vol. 31, no. 11, pp. 1023–31, Nov 2013.

[6] J. S. Berg, M. J. Khoury, and J. P. Evans, "Deploying whole genome sequencing in clinical practice and public health: meeting the challenge one bin at a time," Genet Med, vol. 13, no. 6, pp. 499–504, Jun 2011.

[7] E. R. Mardis, "The $1,000 genome, the $100,000 analysis?" Genome Med, vol. 2, no. 11, p. 84, 2010.

[8] W. F. Fricke and D. A. Rasko, "Bacterial genome sequencing in the clinic: bioinformatic challenges and solutions," Nat Rev Genet, vol. 15, no. 1, pp. 49–55, Jan 2014.

[9] "Hadoop." [Online]. Available: http://hadoop.apache.org

[10] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Sixth Symposium on Operating System Design and Implementation: 2004; San Francisco, CA, 2004.

[11] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). IEEE, 2010, pp. 1–10.

[12] S. Anders, P. T. Pyl, and W. Huber, "Htseq: A python framework to work with high-throughput sequencing data," bioRxiv, 2014. [Online]. Available: http://biorxiv.org/content/early/2014/02/20/002824

[13] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008, pp. 1099–1110.

[14] O. Tange, "Gnu parallel - the command-line power tool," ;login: The USENIX Magazine, vol. 36, no. 1, pp. 42–47, Feb 2011. [Online]. Available: http://www.gnu.org/s/parallel

[15] R. Taylor, "An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics," BMC Bioinformatics, vol. 11, no. Suppl 12, p. S1, 2010.
[16] B. Langmead, K. D. Hansen, and J. T. Leek, "Cloud-scale rna-sequencing differential expression analysis with myrna," Genome Biol, vol. 11, no. 8, p. R83, 2010.
[17] S. Anders, D. J. McCarthy, Y. Chen, M. Okoniewski, G. K. Smyth, W. Huber, and M. D. Robinson, "Count-based differential expression analysis of rna sequencing data using r and bioconductor," Nat Protoc, vol. 8, no. 9, pp. 1765–86, Sep 2013.
[18] A. Schumacher, L. Pireddu, M. Niemenmaa, A. Kallio, E. Korpelainen, G. Zanetti, and K. Heljanko, "Seqpig: simple and scalable scripting for large sequencing data sets in hadoop," Bioinformatics, vol. 30, no. 1, pp. 119–20, Jan 2014.
[19] H. Nordberg, K. Bhatia, K. Wang, and Z. Wang, "Biopig: a hadoop-based analytic toolkit for large-scale sequence data," Bioinformatics, vol. 29, no. 23, pp. 3014–9, Dec 2013.
[20] S. Anders, A. Reyes, and W. Huber, "Detecting differential usage of exons from rna-seq data," Genome Res, vol. 22, no. 10, pp. 2008–17, Oct 2012.
[21] [Online]. Available: http://www-huber.embl.de/users/anders/HTSeq/doc/index.html
[22] [Online]. Available: http://raalesir.github.io/HTSeq-Hadoop/index.html
[23] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin et al., "The sequence alignment/map format and samtools," Bioinformatics, vol. 25, no. 16, pp. 2078–2079, 2009.
[24] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," Genome Biol, vol. 10, no. 3, p. R25, 2009.
[25] "Open Nebula." [Online]. Available: http://opennebula.org
[26] "Cloudera." [Online]. Available: http://www.cloudera.com/content/cloudera/en/why-cloudera/hadoop-and-big-data.html
[27] "Puppet Lab." [Online]. Available: http://puppetlabs.com
[28] S. Schönherr, L. Forer, H. Weissensteiner, F. Kronenberg, G. Specht, and A. Kloss-Brandstatter, "Cloudgene: a graphical execution platform for MapReduce programs on private and public clouds," BMC Bioinformatics, vol. 13, p. 200, 2012.
[29] "Htseq-hadoop," 2014. [Online]. Available: https://github.com/raalesir/Htseq-Hadoop
[30] "Pig latin script for htseqcount." [Online]. Available: https://github.com/raalesir/PigLatinCount

VII. Supplementary Material

- Dataset I: `ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR611/SRR611084/SRR611084.sra`, `ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR611/SRR611085/SRR611085.sra`
- Dataset II: artificial pair-ended dataset for *A.thaliana* (TAIR10) created with the `wgsim` program from the Samtools package.

The reference genome for the mapping was: `ftp://ftp.arabidopsis.org/home/tair/Sequences/whole\_chromosomes/*.fas`

The disproportion between the sizes in Gbases and GB arises since Samtools `wgsim` generates the reads with fixed qualities, so it archives more efficiently than the real data where the qualities differ from nucleotide to nucleotide.

Listing 3: SAM file origin

```
bowtie version 0.12.8, 64-bit
bowtie -S -p 16 a_thaliana -X 600 -m 1
--chunkmbs 500 \
   -1 /dev/fd/63 -2 /dev/fd/62 file.sam
```

TABLE V: Differences in gene counts for Datatet I with the original `htseq-count` and parallelized with `GNU-parallel` (a) and with `HTSeqCount` run on Hadoop (b). The `AT*` are the gene names while the numeric columns are the corresponding count differences. One can see that among 28775 *A. thalina* genes from the feature file `ftp://ftp.arabidopsis.org/Maps/gbrowse_data/TAIR10/TAIR10_GFF3_genes.gff`, there are only 78 small differences in gene counts (a) and 14 (b) with no practical impact on the gene expression analysis result.

GNU-parallel (a)

AT1G04150	1489	1490	AT1G08460	904	905	AT1G16350	1111	1112
AT1G30420	3425	3424	AT1G49870	1076	1077	AT1G50200	3100	3101
AT1G66500	639	638	AT1G69523	542	543	AT1G70320	5356	5357
AT1G71900	1262	1263	AT1G73610	775	776	AT1G75500	964	963
AT1G79800	416	417	AT2G01440	2492	2493	AT2G17290	1340	1341
AT2G26270	1483	1484	AT2G30600	2125	2126	AT2G32960	1615	1616
AT2G33440	1406	1407	AT2G34670	1492	1493	AT2G35635	578	577
AT2G38020	2089	2090	AT2G41680	1377	1376	AT2G42260	668	667
AT2G47970	561	562	AT3G03060	1421	1422	AT3G05420	2160	2161
AT3G12290	996	997	AT3G12900	715	716	AT3G13330	4576	4577
AT3G15470	1475	1476	AT3G19370	1449	1450	AT3G19720	2268	2269
AT3G23660	1881	1882	AT3G41762	198214	198217	AT3G48190	13582	13581
AT3G51090	930	931	AT3G52260	1495	1496	AT3G54300	1299	1300
AT3G56630	882	883	AT3G57650	1671	1672	AT3G59320	891	892
AT3G61210	442	443	AT4G01290	2359	2360	AT4G03200	1516	1517
AT4G14210	2230	2231	AT4G18520	1019	1020	AT4G18640	1563	1564
AT4G24060	980	981	AT4G25560	662	663	AT4G25580	1208	1209
AT4G26580	850	851	AT4G30310	1486	1487	AT4G32120	1328	1327
AT4G33510	948	947	AT4G33520	3078	3077	AT4G38880	791	792
AT5G02250	2268	2269	AT5G13690	2172	2171	AT5G15630	1041	1042
AT5G17830	773	774	AT5G26240	945	946	AT5G27860	852	853
AT5G40480	4070	4069	AT5G45190	1618	1619	AT5G48375	608	609
AT5G48380	983	984	AT5G49020	1605	1606	AT5G51680	832	833
AT5G52100	1142	1143	AT5G53170	2209	2210	AT5G55780	940	941
AT5G59350	748	749	AT5G60350	682	683	AT5G60510	762	761
AT5G60690	2266	2267	AT5G61390	1214	1215	AT5G63480	695	696

HTSeqCount (b)

AT1G30420	3007	3008	AT1G66500	328	329	AT1G75500	774	775
AT2G35635	262	263	AT2G41680	1029	1030	AT2G42260	469	470
AT3G03370	634	635	AT3G48190	13159	13160	AT4G32120	1108	1109
AT4G33510	732	733	AT4G34730	1199	1200	AT5G13690	1757	1758
AT5G40480	3934	3935	AT5G60510	415	416			