1000 Genomes Project Analyzing Chromosome 20

Efthymia Chantzi
Email: Efthymia.Chantzi.0787@student.uu.se
MSc programme in Bioinformatics

Eleftherios Anagnostopoulos
Email: Eleftherios.Anagnostopoulos.8127@student.uu.se
MSc programme in Computer Science

Uppsala University, Sweden

October 6, 2015

Abstract | *Chromosome* 20 *constitutes one of the 22 pairs of autosomal chromosomes in humans. It consists of approximately* 63 *million base pairs and represents almost* 2.5% *of the total DNA in cells. In this work, we analyze the next-generation sequencing (NGS) data for chromosome* 20 *from the "1000 Genomes Project", in order to detect and visualize sequence variation hotspots. We introduce our own distributed and parallel implementation by deploying a Multi-Node Cluster along with the Hadoop Distributed File System (HDFS), which serves as the repository of results. Each DataNode sprawns a number of processes by using multiprocessing in Python, which allows direct, efficient and scalable manipulation of NGS data, stored in highly compressed Binary Format (BAM). Here, we restrict our analysis to paired reads, where the fragment length indicated by the alignment, exceeds* 1000 *base pairs. In this way, we deal with a filtered data set with remarkably reduced size. We focus on tracing alignment matches and single-point mutations; insertions, deletions and single nucleotide polymorphisms (SNPs), among individuals from* 26 *subpopulations, which have been further divided into* 5 *superpopulations*[1] *across Europe, Africa, East Asia, South Asia and the Americas. The greatest average number of alignment matches is pinpointed by Utah Residents with Northern and Western European Ancestry (CEU). Additionally, SNPs comprise the most frequent type of single-point mutations, among all subpopulations and superpopulations, compared to indels. The performance is improved, with the gradual addition of more VMs; from* 1 *to* 4*, and* 2 *processes per VM, which is an indication of a scalable implementation.*

Keywords | 1000 *Genomes Project; Chromosome 20; Sequence variation; Single-point mutations; BAM files; HDFS; Python multiprocessing; Multi-Node Cluster; Fabric*

1. Introduction

The "1000 *Genomes Project*" [2], which dates back to 2008, comprises a remarkable milestone in our understanding of human genetic variation. It is an international research effort to establish the most detailed catalogue of human structural variants from nations all over the world. This kind of genomic information enables researchers to adjust their studies to a wider spectrum of populations and gain more profound knowledge concerning human evolution and phenotypic variation. In this way, unknown but meaningful gene mutations could be unveiled and thus, the roots of many genetic diseases could be pinpointed, potentially paving the way for new and more effective diagnostics and therapeutics.

Next-generation sequencing (NGS), also known as high-throughput sequencing, is a group of different and recent sequencing technologies that allow rapid and significantly less expensive sequencing of DNA and RNA compared to the previously used methods, such as *Sanger* sequencing. The explosion of new NGS platforms and tools is related to the generation of vast amounts of data that require efficient processing and long-term storage. Although the actual size of NGS data varies greatly depending on the depth of sequencing, there are terabytes, if not petabytes of data, that must be stored and analyzed. This work is associated with two major questions; *"How can big and intensive data produced by large-scale sequencing projects be efficiently processed in parallel?"* and *"What kind of differences and similarities can be figured out by looking at variations in the DNA sequences between individual and population samples?"*.

It is true that these questions have already been addressed by a wide variety of scientific approaches using the functional programming philosophy of MapReduce. For instance, the *Genome Analysis Toolkit* (GATK) constitutes a MapReduce framework for analyzing next-generation DNA sequencing data and ensuring robustness, optimization, stability and memory efficiency [1]. Although there exist toolkits, such as GATK, that allow for NGS data analysis pipelines, they do not respond efficiently to parallel access for BAM files. This is where the contribution of the *Hadoop-BAM* Java library lies. It comprises an integration layer between analysis applications and BAM files stored in the HDFS by offering a convenient API for implementing map and reduce functions for this kind

[1] http://www.1000genomes.org/category/frequently-asked-questions/population, accessed: June 1, 2015
[2] http://www.1000genomes.org/, accessed: June 3, 2015

of NGS stored data [2].

Here, we deal with chromosome 20 and specifically, around 1TB of NGS data in BAM format, provided by the "1000 Genomes Project". This format is the binary version of the text-formatted files in SAM format and is designed to compress reasonably well. Consequently, less archiving space is required for the same information. However, their processing and parsing are differentiated. We present two approaches; our initial approach, and our proposed final solution to the encountered problems. Our analysis aims at filtering the initial data set and keeping only the aligned paired reads, where the template length exceeds 1000 base pairs. The greater the template length, the more likely it is to point out structural genetic variation. We investigate the number of alignment matches, soft-clipped regions, single-point mutations; indels and SNPs, per individual, as well as the average number of all these structural patterns among the 26 included different subpopulations.

2. Materials and Methods

Our implementation consists of two major and indispensable parts; *filtering* of the initial data set and post-filtering of the resulting filtered data set, in order to detect and visualize hotspots of human genetic variation, lying on chromosome 20. It is important to mention that both these aspects of our implementation take place under the *MapReduce* programming model. The adopted programming model is structured by an architecture design of 4 virtual machines (VMs); 1 Master and 3 Slave nodes, which are build on the environment of *Lab2-base-image*. The programming language used for the development of our implementation is *Python*.

2.1. Filtering

As already stated, the initial size of our data set is almost 1TB. By implementing the appropriate filtering and excluding all paired reads with fragment length less than 1000 base pairs, the size is reduced by a factor of 1000; $\sim$ 1GB. Provided the specifications of our architecture design, which is thoroughly presented in the next subsection, we were not able to use the whole initial dataset. We have used instead a considerable subset of approximately 80GB, which is spread out across the 4 deployed VMs. It should be stated that we have included 6 individuals from all the subpopulations of the whole data set, because we consider it important to maintain the wide geographical scale, even at the expense of less individuals. The resulting filtered data set is estimated to occupy 625KB of space.

We use *Pysam* [3], which is a python module for parsing of SAM/BAM files. More precisely, we focus on three different properties; *template_length* for excluding tuples with fragment length less than 1000 base pairs, *cigartuples* for getting the Compact Idiosyncratic Gapped Alignment Report (CIGAR) string and *MD tag* for distinguishing between the alignment and sequence matches. Before proceeding with the exact procedure of filtering, we provide more detailed information regarding the *CIGAR string* and *MD tag*. The former is used in order to retrieve the number of alignment matches, insertions, deletions and soft clips; unaligned regions that are included in the sequence. As far as the latter one is concerned, it is used, so as to specify the number of sequence matches and thus, estimate the number of SNPs. Consequently, these two properties allow us to detect similarities and structural variants.

The described process of filtering pertains to each one of the individuals. More precisely, for each BAM file all reads with *template_length* less than 1000 base pairs are excluded. Then, the total number of alignment matches, indels, SNPs and soft clips are estimated. The results are saved in one *.txt* file with the following format:

```
Individual.Country.Deletions    numOfDeletions
Individual.Country.Insertions   numOfInsetions
Individual.Country.Matches      numOfMatches
Individual.Country.SNPs         numOfSNPs
Inidvidual.Country.Soft_clips   numOfSoft_clips
```

The purpose of using this kind of format is that the output can be processed with *Hadoop MapReduce*.

2.2. Initial Approach

Our initial approach included three different stages; uploading of the input data to *HDFS*, use of *Hadoop streaming* for parallel processing and use of *Pysam* for file parsing. However, *Hadoop streaming* reads data using stdin and *Pysam* reads BAM files locally, or using *Python streaming*. We were unable to connect them. At this point, it should be mentioned that we had already implemented the filtering function using *Pysam*, so we did not end up with other existent alternative libraries, such as *Hadoop-BAM* [2] or *ADAM* [4].

2.3. Proposed Solution

Our proposed solution was to deploy a *Multi-Node Cluster*, where each *DataNode* processes a part of the dataset *locally* and uploads results to *HDFS*. The underlying idea is that each DataNode spawns a number of processes, using *multiprocessing* in Python. It is also significant to mention that the *NameNode* uses *Fabric* [5] for deploying parallel tasks on each *DataNode*. Figure (1) illustrates the architecture design that we have adopted.

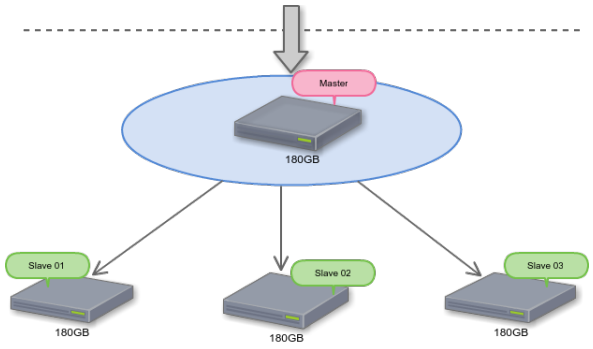

Figure 1: *Architecture design:* 4 *VMs* $\longmapsto$ 1 *Master and* 3 *Slaves*

The *NameNode* receives a text file as input, containing the names of the BAM files that need to be processed. The file is

then splitted and distributed to *DataNodes*, so as each *DataNode* can process a smaller part of the input data set. Each *DataNode* downloads the corresponding files and stores them locally. Thus, the *NameNode* can initiate a number of parallel tasks, where each *DataNode* applies the filtering function to its corresponding part of the data set and uploads the results to *HDFS*. In this way, two levels of parallelism are achieved, since multiple *DataNodes* run filtering tasks in parallel, where various processes can be spawned, in order to execute each task.

2.4. Post-filtering

Post-filtering pertains to the parallel processing of the results gained by the filtering step. Once, the results for each individual have been generated and stored into *HDFS*, they are available for further investigation. They are in simple text format, which renders their further analysis convenient and flexible. We have chosen to apply a second *MapReduce* programming model, in order to convert the individuals' results into subpopulations' results. Being more specific, we represent the average number of alignment matches, indels, SNPs and soft clips with reference to the originating subpopulation among the 26 different in total. The format of the post-filtering results is identical to those of the filtering procedure:

```
Country.Deletions     averageNumOfDeletions
Country.Insertions    averageNumOfInsetions
Country.Matches       averageNumOfMatches
Country.SNPs          averageNumOfSNPs
Country.Soft_clips    averageNumOfSoft_clips
```

This step can be adjusted to the needs and requirements of the current analysis. For the time being, we consider important to examine patterns of genetic variation among subpopulations. However, this does not mean that a more general analysis of the superpopulations would be either impossible or out of scope. Actually, this would be very useful and wise in case we were trying to map structural variants with known and prevalent phenotypic traits that are determined by specific genes of chromosome 20, in a wider geographic scale. The initial results per individual from the filtering step, can be stored for long period of times, owing to their flexible *.txt* format and considerably small size; 625KB. Consequently, the analysis can be easily adjusted by the user to the current needs and requirements.

3. Results

3.1. Analysis

In this work, we have gained two sets of results by applying two separate *MapReduce* models for each one of them. More precisely, the first set pertains to the total number of alignment matches, indels, SNPs and soft clips per individual. This filtered data set has been generated by the filtering of the initial subset ($\sim$ 80GB) of BAM files. The second set of results has been produced by the first one, so as to categorize and organize the results according to the originating subpopulation. We provide five different bar charts; one for each of the 5 patterns of structural similarity or difference that we have adopted. The bar charts obtained by the *post-filtering analysis* are attached to Appendix A. There are in total *five* bar charts, including the average number of deletions, insertions, alignment matches; they can be either sequence matches or mismatches, single-nucleotide polymorphisms; SNPs and soft clips; unaligned regions that are present in the sequence, among the 26 different subpopulations.

3.2. Performance & Scalability

At this point, we measure the performance and scalability of our implementation. It is important to ensure that the deployed implementation is able to scale, when more machines are added to the pool of resources. For this reason, two diagrams, regarding the average running time of the filtering step, are attached to Appendix B. The first one illustrates the average running time for the filtering step, when 1, 2, 3, 4 VMs are deployed with 1 and 2 processes per VM. As far as the second figure is concerned, it depicts the gain in performance, when more VMs and processes per VM are added. For this type of measurement, the initial running time with a specific number of VMs and processes is measured and then, this quantity is divided with the running time arising from the increased number of VMs and processes, indicating the observed patterns of speeding up.

4. Discussion

It is true that several useful conclusions have been reached by the fulfillment of this work. Firstly, the significance of understanding the given problem and deploying the appropriate implementation has been indicated. It is very important to be able to examine the problem thoroughly, which is the format of the given data in this case, and choose the architecture that best suits the requirements of the respective analysis; efficient *parallel processing* and *scalability*.

4.1. Analysis

As far as the post-filtering results are concerned (Appendix A), several meaningful patterns of human structural variation have been figured out, with reference to the originating subpopulation. It is interesting that Utah residents with Northern and Western European Ancestry (CEU), who belong to the superpopulation of Europeans (EUR), appear to dominate the rest of the subpopulations, in all categories. They have been indentified with the greatest average portions of deletions, insertions, alignment matches, SNPs and soft clips, as well. From an evolutionary point of view, this could be an indicator of higher mutation rate, which leads to higher evolutionary rate in comparison with the reference genome. However, such deviations may indicate causal links or pre-disposing factors for specific diseases that are more prevalent among this subpopulation. Additionally, Gambians in Western divisions in the Gambia (GWK) and Luhyas in Webuye, Kenya (LWK), both from Africa, show the most prevalent patterns after CEU in all cases. Consequently, one subpopulation from Europe and two

from Africa indicated the most alignment matches and single-point mutations.

Moreover, by looking carefully at the obtained bar charts from the post-filtering analysis, it is apparent that the single-nucleotide polymorphisms prevail in all subpopulations compared to insertions and deletions. Therefore, it could be concluded that it is much more likely and frequent for a DNA character to be substituted with an other one, rather than a new character to be inserted in a position or an old character to be deleted from the sequence. The greatest average number of insertions and deletions is specified among the subpopulation of CEU, while the smallest among the Gujarati Indians from Houston, Texas (GIH). The same pattern is observed for SNPs, but with remarkably larger average numbers for each category. Lastly, soft-clipped regions are dominant among CEU, GWD and YRI subpopulations, whereas they represent a relatively small amount of genomic regions in the GIH subpopulation.

4.2. Performance & Scalability

Judging from our measurements, we reach the conclusion that the performance of the proposed solution is improved as the number of used *DataNodes* is increased. On the other hand, this does not mean that the architecture of the implementation is irrelevant to the size of the input dataset. For example, a Multi-Node Cluster consisted by 100 virtual machines, for the processing of 100MB of data, should undoubtedly be considered as inefficient. Furthermore, the used architecture should also be dependent on the available resources. In our case, where the *NameNode* uses *Fabric* for deploying parallel tasks on each *DataNode*, the level of parallelism is limited by the number of the cores on the *NameNode*, as the parallel functionality of *Fabric* is implemented via the Python multiprocessing module. For this reason, a solution, where the number of used *DataNodes* is much greater than the number of cores on the *NameNode* might not be productive. Finally, the available resources and the size of the dataset should also be taken into account while choosing the number of processes running on each *DataNode*, since additional workers might lead to delayed execution, owing to slower startup times and process synchronization.

5. Future Work

Although this work is associated with several meaningful findings, it is true that it was completed in a short period of time and simultaneously, implementation problems were encountered at the beginning. Therefore, there might be several improvements and extensions that would be of the utmost importance for a project of this kind. Taking all these into consideration, we are thinking of a forthcoming and more detailed analysis at the level of filtering and post-filtering that would incorporate the exact positions of the found structural variants. Provided the positions of the genes that are located on chromosome 20 and their related diseases and disorders, we could elaborate on population-linked diseases. This kind of information would be very useful for geneticists, who try to specify the inheritance of genetic diseases and improve diagnostics and therapeutics.

From a technical point of view, it would be great, if we could test our implementation in the total data set of BAM files. What is more, it would make sense to experiment with similar data sets of other autosomal or even sex chromosomes. In this way, we would be able to gain a more reliable overview of the achieved scalability.

References

[1] A. McKenna *et al.*, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data," *Genome Res.*, vol. 20, pp. 1297–1303, Sep 2010.

[2] M. Niemenmaa *et al.*, "Hadoop-BAM: directly manipulating next generation sequencing data in the cloud," *Bioinformatics*, vol. 28, pp. 876–877, Mar 2012.

[3] "pysam - An interface for reading and writing sam files." http://pysam.readthedocs.org/en/latest/api.html. Accessed: June 3, 2014.

[4] M. Massie *et al.*, "Adam: Genomics Formats and Processing Patterns for Cloud Scale Computing," Tech. Rep. UCB/EECS-2013-207, EECS Department, University of California, Berkeley, Dec 2013.

[5] "Welcome to Fabric." http://www.fabfile.org/. Accessed: June 3, 2014.

GitHub Repository Link

https://github.com/EffieChantzi/LDSA

A. Appendix A

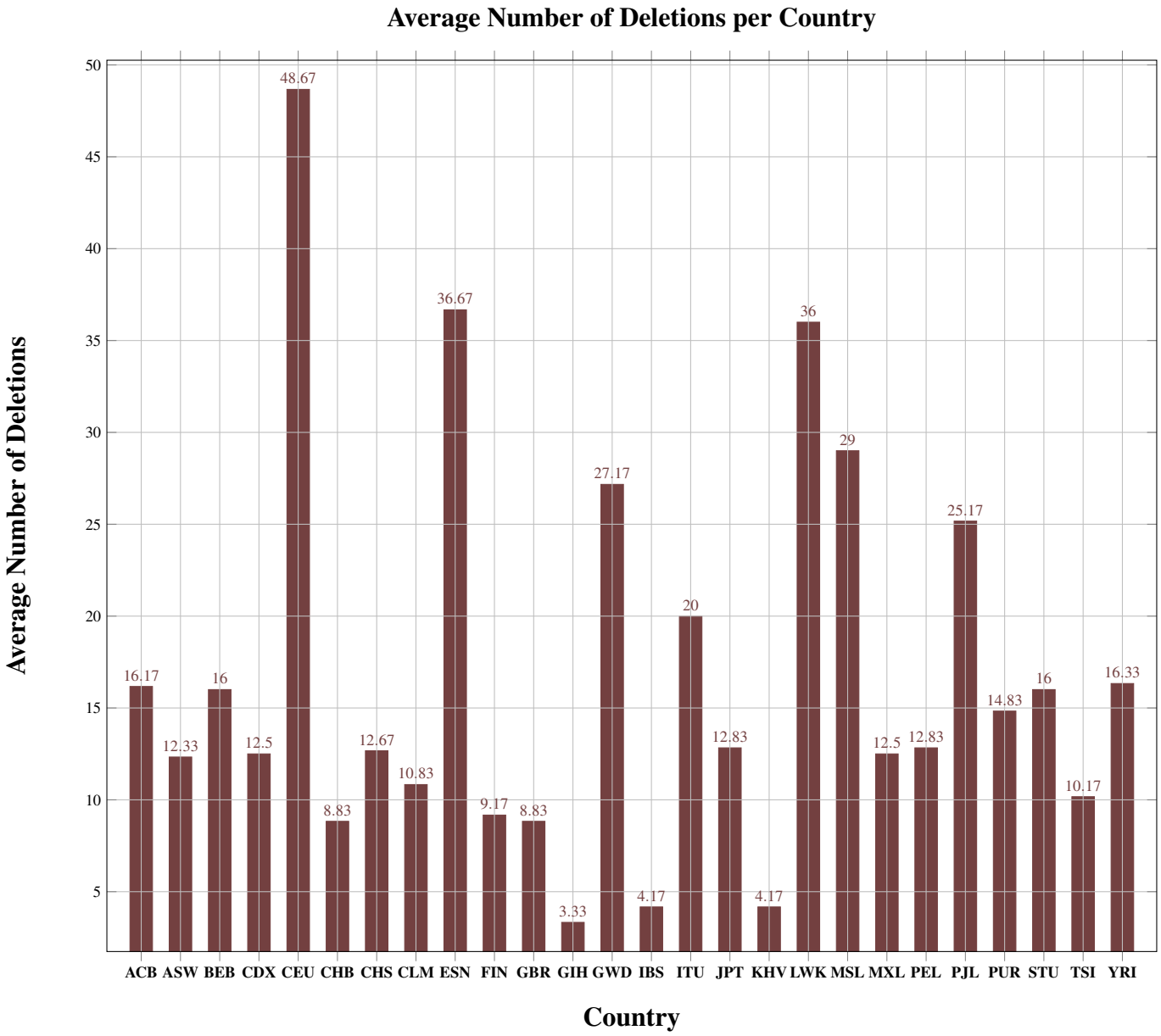

Figure 2: *Post-Filtering Results: Average number of Deletions, using 6 individuals per country.*

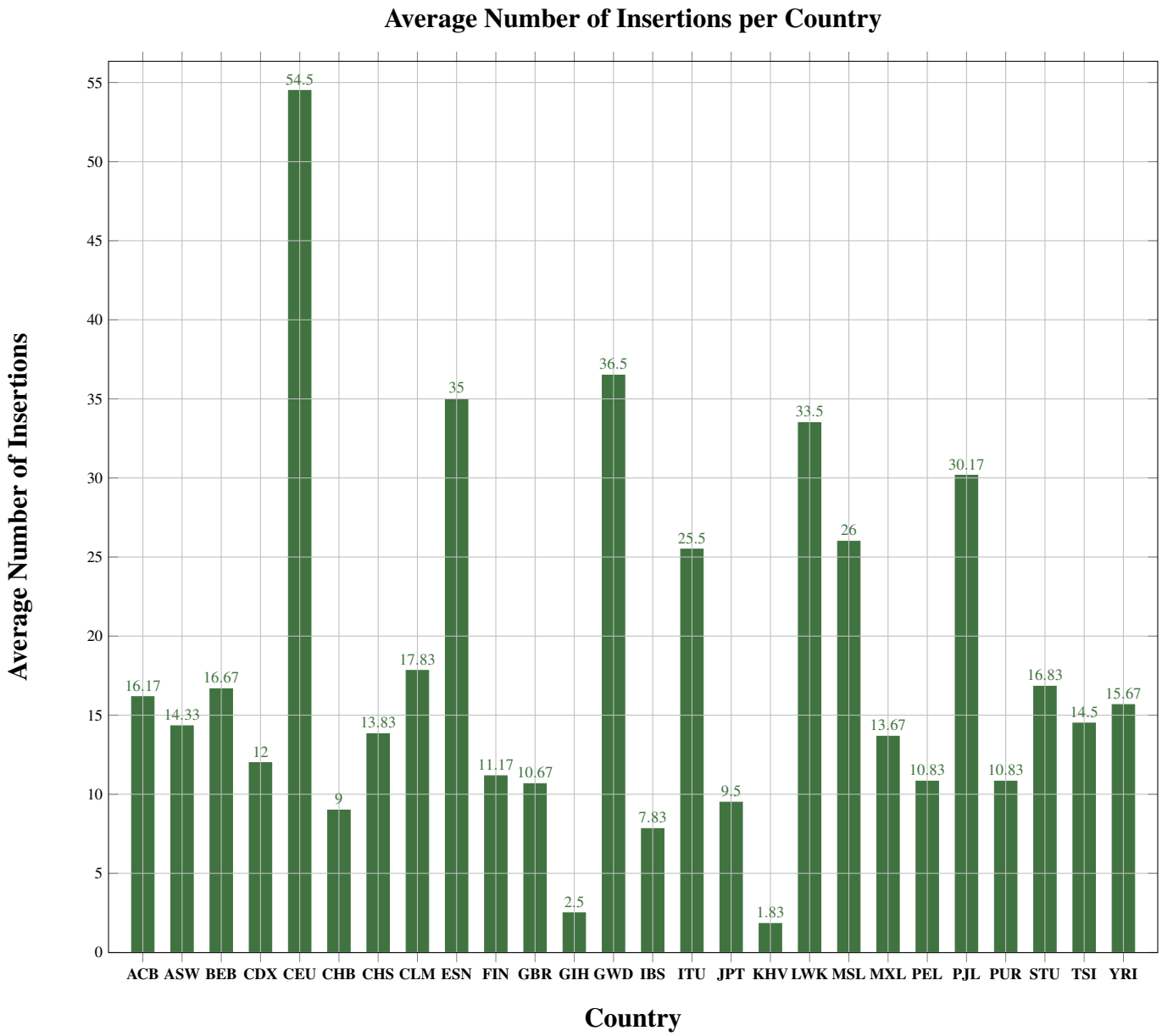

Figure 3: *Post-Filtering Results: Average number of Insertions, using 6 individuals per country.*

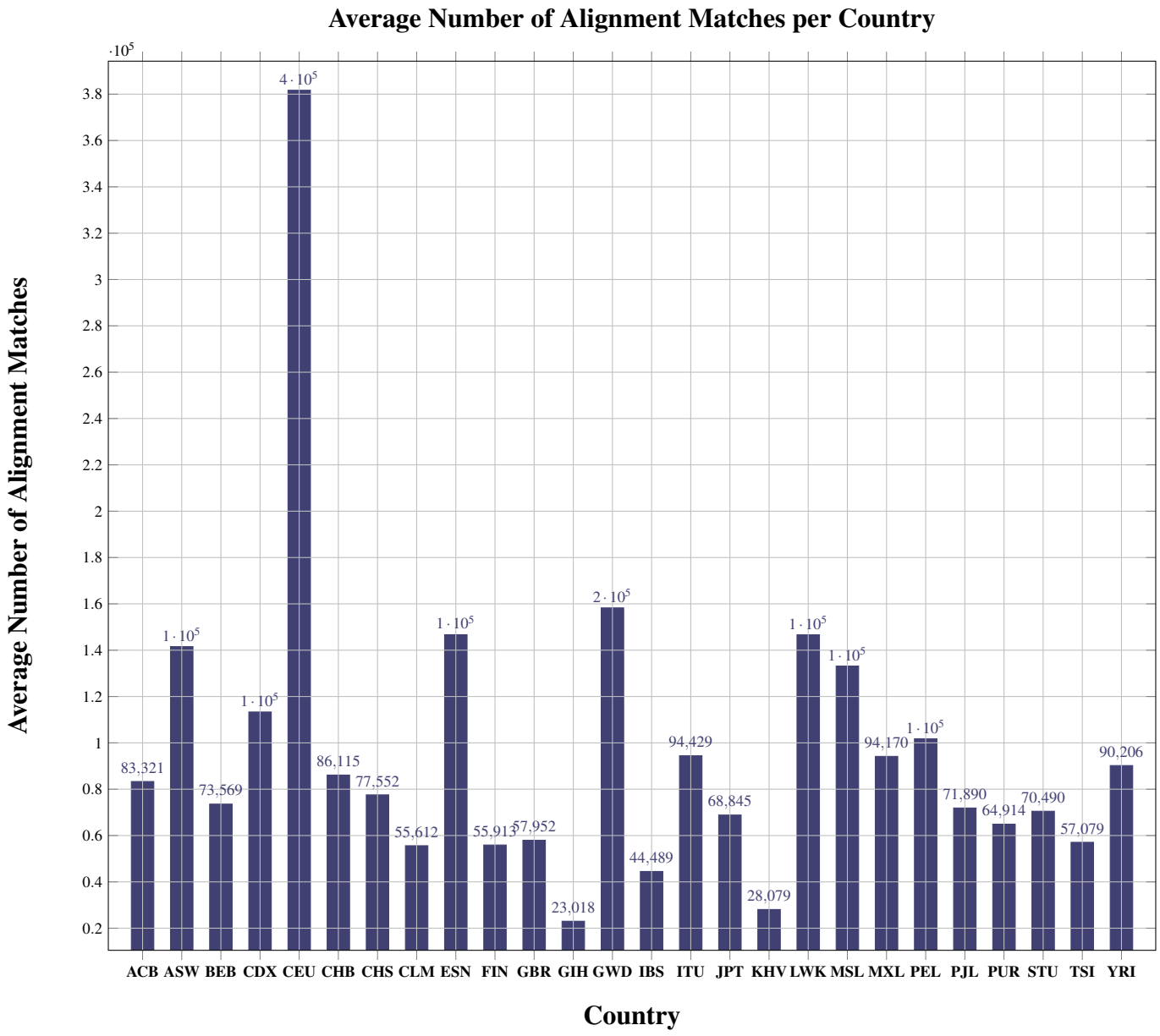

Figure 4: *Post-Filtering Results: Average number of Matches, using 6 individuals per country.*

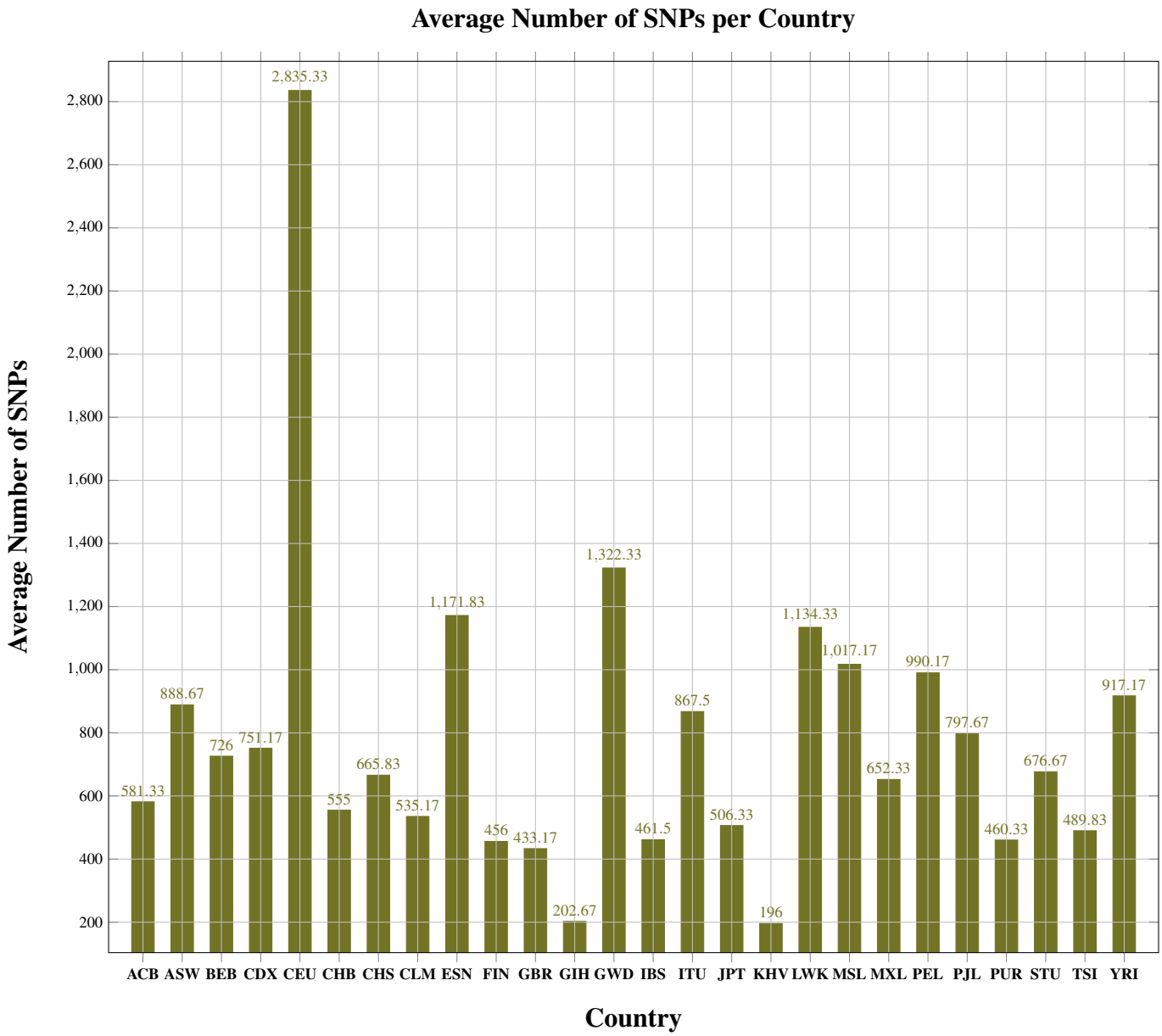

Figure 5: *Post-Filtering Results: Average number of SNPs, using 6 individuals per country.*

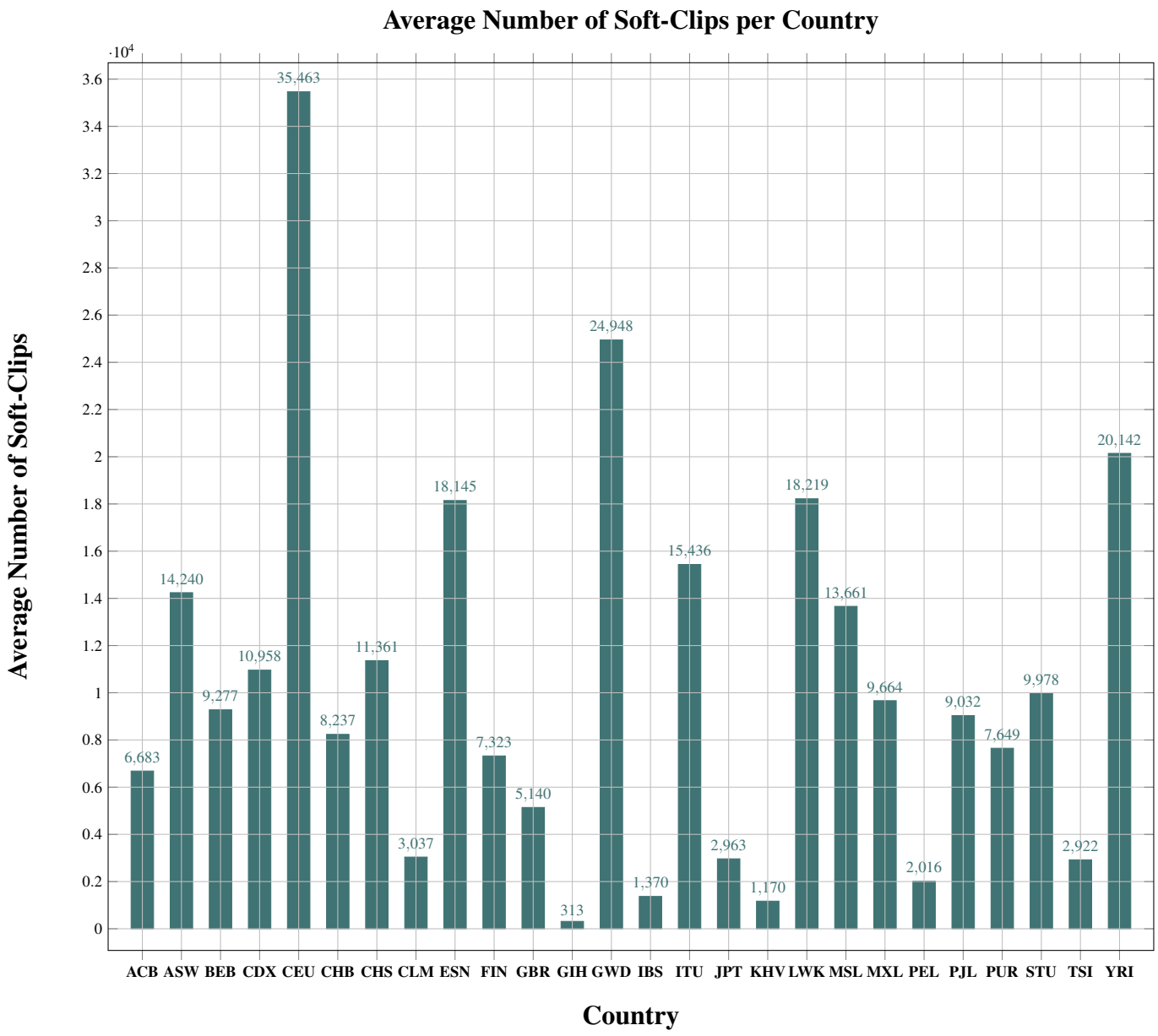

Figure 6: *Post-Filtering Results: Average number of Soft-Clips, using 6 individuals per country.*

B. Appendix B

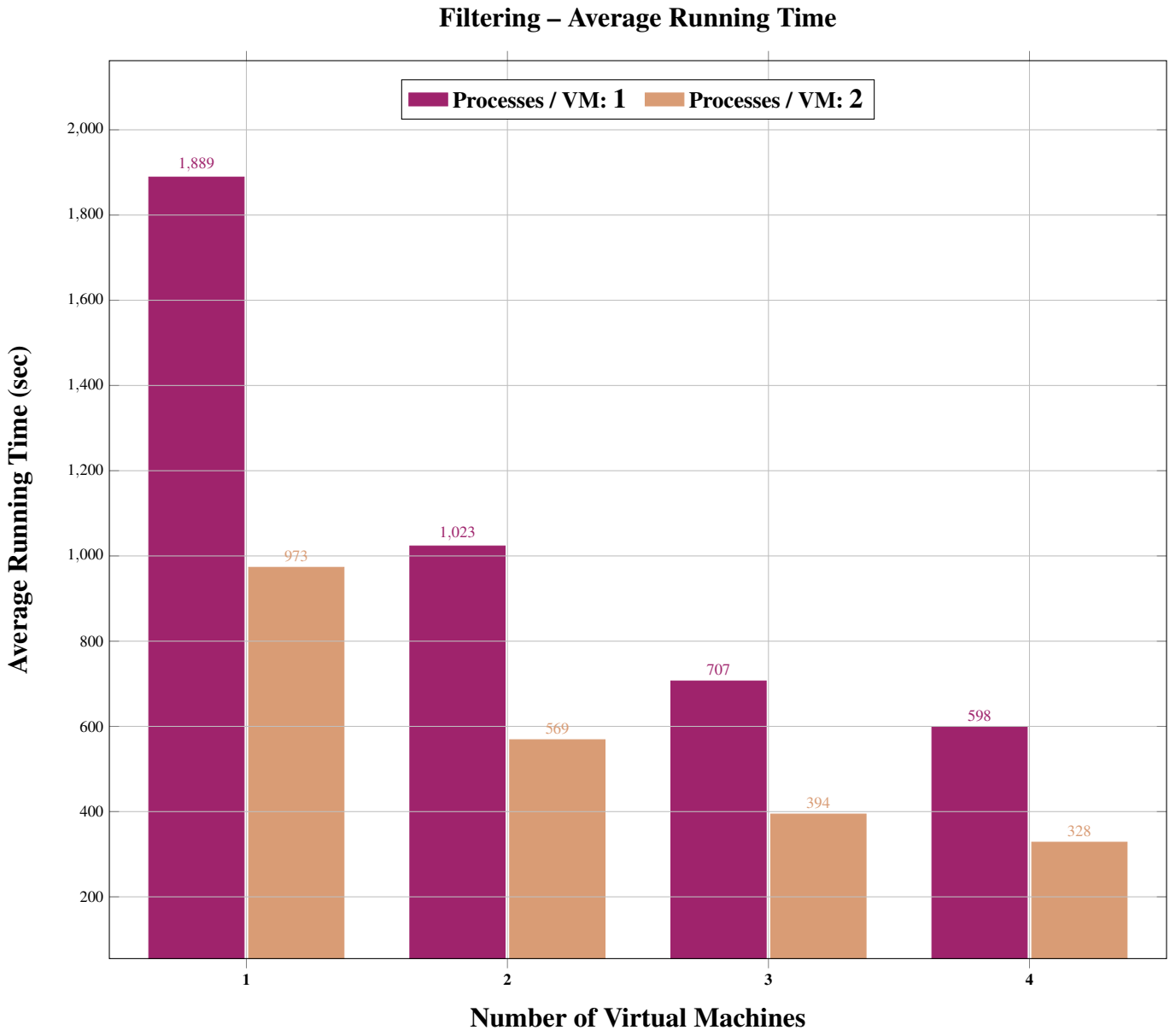

Figure 7: *Filtering Results: Average running time using various numbers of virtual machines and processes, over* $\sim$ 80*GB of data.*

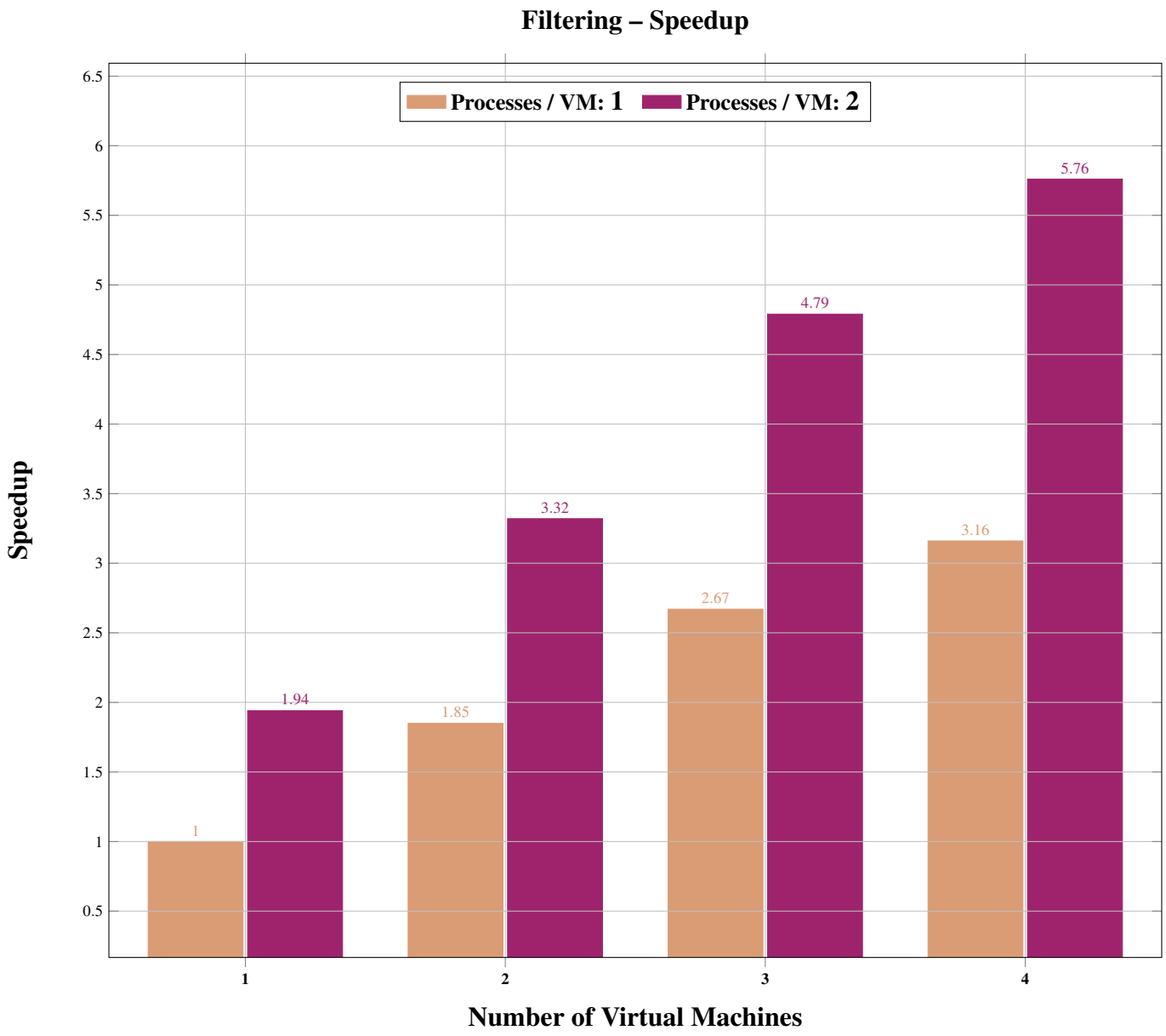

Figure 8: *Filtering Results: Speedup values using various numbers of virtual machines and processes, over* $\sim$ 80*GB of data.*