# A quantitative assessment of the Hadoop framework for analyzing massively parallel DNA sequencing data

Alexey Siretskiy[1*], Luca Pireddu[2], Ola Spjuth[3,4]
**1 Department of Information Technology, Uppsala University, Uppsala, Sweden**
**2 CRS4, Polaris, Pula, Italy**
**3 Department of Pharmaceutical Biosciences, Uppsala University, Uppsala, Sweden**
**4 Science for Life Laboratory, Uppsala University, Uppsala, Sweden**
**∗ E-mail: alexey.siretskiy@it.uu.se**

## Abstract

New high-throughput technologies such as massively parallel sequencing has transformed the life sciences into a data-intensive field. With increasing data volumes comes the necessity to analyzee data in parallel using high-performance computing resources, but doing this effectively can be laborious and challenging. Hadoop, emerging in the last decade, is a framework that automatically distributes data and computation and has been shown to scale to thousands of nodes. Herein we introduce metrics and report a quantitative comparison of Hadoop to regular high-performance computing resources for aligning short reads and calling variants for five datasets of different sizes up to 250 gigabases. In order to increase performance of existing software and obtain a better comparison we modified and wrote new analysis scripts. From the observed scaling relations we are able to draw conclusions about the perspectives of the approaches, leading to the conclusion that as data set sizes reach 100 gigabases, the Hadoop-based pipelines become performance-competitive with a canonical high-performance cluster solution. As data sets in biological sequencing are sure to increase with time, Hadoop and similar frameworks are very interesting technologies that we envision will play a key role in the future of biological data analysis.

## Author Summary

New experimental technologies generate vast amounts of data and require supercomputers to store and analyze. As datasets increase in volume there is a need for new computational tools and technologies that can cope with the scale of data, and in this paper we provide a detailed study comparing the performance of the Hadoop framework with traditional high-performance computing for data in biological sequencing. While there is an overhead of using Hadoop, we show that when sequencing data sets reach sizes of 100 gigabases then Hadoop becomes attractive compared with traditional methods. We also show that Hadoop scales linearly while traditional systems become network-saturated at larger data sizes. As sequencing data are bound to increase in the future, Hadoop is a very interesting framework to base future analysis on.

## Introduction

Since its inception, massively parallel DNA sequencing, also referred to as Next Generation Sequencing (NGS) technology, has been an extremely bountiful source of data giving insight into the workings of biological machinery [1,2]. Decreasing sequencing costs facilitates and promotes larger and larger studies with increasingly larger data sizes, and extracting useful information from these voluminous amounts of data is transforming biology into a data-intensive discipline that requires a high-performance computing (HPC) infrastructure to analyze. As an example of the scale of the demands, consider that a single Illumina high-throughput sequencing run produces approximately 1800 gigabases (Gbases) corresponding to 2 terabytes (TB) of raw data in 3 days [3].

A common step of NGS data analysis consists of alignment short reads to a reference sequence and then finding the genetic variations specific to the sample. Many of the most widespread tools for this purpose, like BWA [4], Bowtie [5] and Samtools [6], are not made with distributed computing in mind. Many others do not even have the native ability to use multiple cores.

The most common approach to speed up NGS tools is to parallelise within a compute node (multi-core parallelisation) using shared memory parallelism (OMP) [7], but this approach is naturally limited by the number of cores per node, which does not usually exceed 16, see [8]. For the tools which do not support OMP natively, e.g. Samtools, variant calling can be parallelised by creating a separate process for each chromosome, or using GNU Parallel Linux utility [9]. Of great importance is that a multi-core approach does not improve the performance of operations that are limited by local disk or network throughputs, whereas splitting the dataset and multi-node parallelisation generally are not bound by these constraints.

Message Passing Interface (MPI) [10] is a common way to implement multi-node parallelisation, but writing efficient MPI-programs for hundreds of cores is a non-trivial task since thread synchronisation (or load balancing) has to be woven into the software code and there are few existing solutions available for processing sequencing data [11–13].

Another common way to introduce parallelisation to NGS analysis pipelines in Linux systems is to use Bash scripting. This involves using existing utilities and cluster tools to split the data into chunks, process them on the separate nodes, and merge the results afterwards. This kind of solution benefits from both MPI-like and OMP parallelisation and provides good performance, but the development requires substantial expertise in order to be efficient. Since the process is tightly coupled to the local computational cluster and network architectures, it might also not be possible to be re-used in other settings.

The Map-Reduce (MR) programming paradigm [14] offers a compelling alternative for running tasks in a *massively* parallel way. This paradigm, however, shifts the focus from the best performance to scalability, suited for managing huge datasets of sizes up to several terabytes [15]. The most prevalent open source implementation of Map-Reduce is Hadoop [14, 16]. The Hadoop MR framework provides automatic distribution of computations over many nodes as well as automatic failure recovery (failure of individual jobs or computing nodes by storing multiple copies on different nodes), and automated collection of results [16]. Hadoop Distributed File System (HDFS) is a complementary component that stores data by automatically distributing it over the entire cluster, writing data blocks onto the local disk of each node and therefore effectively moving the computation to the data and reducing network traffic. HDFS provides a storage system whose bandwidth and size scales with the number of nodes in the cluster [17], which is very different from the properties of the usual HPC cluster network architecture.

In this manuscript we focus on the question *when* Hadoop is an appealing alternative to the program tools generally found in HPC centers for DNA-seq analysis. Since Hadoop is written in Java, which is slower than the standard HPC programming languages like C or Fortran, we seek to estimate an average data size when it starts to be worthwhile to use Hadoop from a performance perspective. We also study execution times for datasets of different sizes and analyze Hadoop and HPC approaches from a scaling perspective.

## Results

We constructed two pipelines for identifying single-nucleotide polymorphisms (SNPs) from short read data, one based on Hadoop and the other on regular HPC with a batch processing system (hereafter referred to as HPC method) (Methods). Our experiments then consisted of running the pipelines for five datasets with short reads of different sizes and measuring the wall-clock run time for each pipeline stage. All experiments were repeated several times and the results averaged to obtain a data point for the particular combination of data size and computational platform. For the existing Hadoop software we propose modifications, in order to benefit fully from massively parallel nature of MR computations. For the classical HPC DNA-seq analysis programs we developed a set of Bash scripts utilizing multiple nodes for short read alignment and exploiting the network fully, thereby speeding up calculations.

## Modified preprocessing stage

A native preprocessing stage in Crossbow provides great flexibility in delivering the data to the Hadoop cluster, where data can be downloaded from Amazon S3, FTP and HTTP servers over the Internet [18]. The only way to introduce parallelisation in the native preprocessing stage is to split the read files into smaller chunks and to generate a manifest file listing all these chunks. This, however, is lacking the massively parallel way of treating the data.

Assuming that sequencing platforms are usually in close proximity with facilities that provide both storage and computers, we considered that the data already had been downloaded to storage that Hadoop could access. We rewrote the Crossbow preprocessing stage scripts in a MR-manner, benefiting from its massively parallel nature. The requirement is that the FASTQ data are i.e. BZIP2 archived (however, any splittable archive format is suitable) and accessible by Hadoop. For our case the storage with the data was mounted with SSHFS to one of the Hadoop nodes, from where the BZIP2'ed data were transferred to the HDFS. BZIP2 provides very good compression and is splittable, meaning that the archive can be expanded in a parallel manner. Also, BZIP2 format is natively supported by Hadoop, i.e., there is no difference in dealing with the FASTQ data or with its BZIP2 archive.

The Hadoop streaming library offers a possibility to write code for Hadoop jobs in any programming language. Our Python scripts efficiently process short reads, produced by Illumina sequencing platform of different versions, as well as the FASTQ files converted from SRA format (NCBI Sequence Read Archive [19]). The problem of lacking the unique FASTQ header standard was solved by the rewriting the header based on a SHA-1 hash function, and the reads mates were labeled with ".1" and ".2" suffixes. To ensure that both the forward and reverse reads of the same pair would end up on the same Reducer, secondary Mapper key-sorting mechanisms were involved.

In order to compare the native preprocessor and the proposed one, the data was put on HDFS beforehand to eliminate possible network delays, e.g., while downloading data from Amazon. Table 2 illustrates the benefits of our approach. To complete the comparison the same preprocessing was performed with a BASH script for the same data located on the HPC storage. The script utilizes multiple cores and was executed on a single HPC node (see Supplementary information).

## Accuracy of pipelines

Since our HPC and Hadoop approaches use different SNP callers (Samtools and SOAPsnp, correspondingly) we should not expect them to deliver perfectly matching SNP lists, but still we expect them to capture and correctly identify the mutations. We tested the correctness just for the smallest dataset (I). The mutation $C \rightarrow T$ on chromosome 4 at position 16702262 [20] was successfully localized by both applications.

## Scalability of HPC and Hadoop approaches

To demonstrate the scalability of the HPC approach we collected running times for dataset I as a function of the number of cores used (Table 3). The alignment process in Bowtie scales fairly well, but the SNP calling part in Samtools is a bottleneck, scaling worse than Bowtie, and consuming a progressively larger portion of the total calculation time even after parallelizing analysis by chromosome as previously suggested [21]. For the given datasets (I–V) the timings for alignment against the corresponding genomes and SNP calling for the HPC and Hadoop approaches were collected (Table 4).

One of the attractive sides to run i.e. Samtools, Bowtie, etc. in the Hadoop MR framework is its almost linear scalability, i.e., calculation time linearly depends on the size of the dataset [18,22], regardless the scaling nature of the underlying program. Figure 1, based on Table 4, shows the calculation time as a function of the dataset size for the 56 cores Hadoop cluster and datasets I-IV.

## Comparing Hadoop and HPC runtime efficiency

Hadoop was designed to digest huge datasets [14,15]. One can propose then that the larger the dataset is, the more suitable Hadoop becomes compared to the HPC approach. In order to compare the "suitability" for different types of calculation platforms (Hadoop and HPC) each being run on a different number of cores, we constructed the function showing the amount of used core-hours: $F = T_p \times p$, where $T_p$ is the calculation time on $p$ cores.

Using the data from Table 4, we plot the ratio $F_{Hadoop}/F_{HPC}$ as a function of the reciprocal dataset size (Figure 2). Extrapolation to zero on the $X$-axis estimates the ratio for the hypothetical infinite dataset. As one can see, the Hadoop approach becomes more and more effective compared to the HPC scenario as the dataset size increases. The parabolic extrapolation for our settings give the ratio as $1.70 \pm 0.01$, meaning that Hadoop running even in the virtualized environment of a private cloud assembled on moderate hardware, is competitive with the HPC approach run on "bare metal" of modern hardware for datasets greater than 100 Gbases (dataset IV), which is typical for human genome sequencing with a sequencing depth of about $30x$.

The simplest curve to fit the points nicely is a parabolic function, importantly not a constant. Since the Hadoop approach for the datasets reveals the linear scaling (Figure 1), a deviation of the ratio from the constant shows the lack of scalability of the HPC approach. We suspect the main reason for such a behavior is the fact that for large datasets (dataset III-V in our case) the SNP calling routine with Samtools becomes more memory and time demanding than the actual alignment with Bowtie. If lacking RAM, Samtools ~~implementation~~ swaps to disk, creating multiple (up to hundreds) temporary files while sorting the BAM file, keeping the network load for NFS storage and time-expensive IO on a high rate. Table 1 shows the sorting runtime performance of Samtools for the dataset IV run with different amount of RAM available on a single HPC node. The large uncertainty for small amounts of RAM we believe is due to other users' load on the cluster network. As a result, a large amount of RAM per node is needed in order to use Samtools effectively, while Hadoop accomplishes SNP calling on much more economic hardware in linear time with just 12GB RAM per virtual node.

## Comparing the network communication efficiency for the Hadoop and HPC approaches

The network communication model for Hadoop has a major difference from the usual HPC cluster network architecture ($n$-tier tree) with NAS or NFS attached storages. The effective bandwidth of the Hadoop network increases with the cluster size [17], opposite to that of the HPC network where cluster growth results in network saturation and performance depletion. To study this phenomena, we compared HPC and Hadoop network communication costs depending on the number of nodes involved for a fixed dataset size (58 Gbases, dataset IV).

Due to the trivially parallelisability of the alignment process – the read-pairs are independent of each other, and can be aligned independently – one could try to involve more computational resources, e.g., split the initial data into chunks to process them independently. Reducing the size of each data chunk reduces the aligner job, $T_{alignment}$, but at the same time, the more chunks almost simultaneously have to be sent over the network, potentially causing traffic jams, and therefore increasing the communication costs, $T_{comm}$.

There are several program packages for short read alignment with MPI support [11,13], and almost linear scaling up to 32 nodes for pair-ended reads has been reported [23]. However, e.g., the Pmap package functions poorly for datasets larger than 20 Gbases. To circumvent this limitation we implemented a highly optimized Bash script making use of standard Unix utilities to use the HPC cluster network as efficiently as possible [24], and compared the network performance with the standard Hadoop HDFS approach. We separated the alignment time, $T_{alignment}$, and the communication time, $T_{comm}$, and plotted $T_{alignment}/T_{comm}$ as a function of the reciprocal number of nodes $1/N$ (Figure 4). This measure is applicable to both HPC and Hadoop, however, $T_{comm}$ has a different explanation. For Hadoop, the short reads in FASTQ format have to be preprocessed (involving node communication $T_{comm}$) to be able to run in MR-fashion, while the data locality will be automatically achieved during the data ingestion into the HDFS. We rewrote the code for the preprocessing stage for Crossbow to make it suitable for MR-style parallelisation. For the HPC approach, $T_{comm}$ involves the chunks that are sent from the sequence delivery location to the local node scratch disks where the actual alignment happens, and the sending of the aligned SAM files back to the delivery location over the network.

*Hadoop results (filled circles in Figure 4)*: One can see that the ratio $T_{alignment}/T_{comm}$ reveals very weak dependency in a wide range of number of nodes $N$: from 4 up to 40. It is known that Bowtie provides almost linear scaling between alignment time and dataset chunk size $D$: $T_{alignment} \propto D \propto 1/N$, see [5], and Figure 1. Since the ratio $T_{alignment}/T_{comm}$ is approximately constant, one can conclude that $T_{comm} \propto 1/N$, meaning that the more nodes involved, the faster the communication is in the preprocessing stage.

*HPC results (open circles in Figure 4)*: The data from the delivery location is split into chunks in parallel, which are simultaneously pushed to the local scratch disks of the nodes allocated by SLURM. One can see two distinct linear stretches. One stretch is for the range from 4 to about 12 nodes, and the another is from 12 up to 60. The former (horizontal stretch) is explained as for Hadoop – the more nodes involved, the faster the chunks are being distributed. The latter stretch with the positive slope could be explained as follows: In the region of about 12 nodes the network becomes saturated and unable to pass more data in a time unit, while the alignment time is still proportional to the chunk size: $T_{comm} \approx \text{const}, T_{alignment} \propto D \propto 1/N \rightarrow T_{alignment}/T_{comm} \propto 1/N$, i.e., a linear dependency, which can be observed in the plot. The transition area between the two modes is based on the fact that each hard disk drive on the local node can write the data at a speed of about 1Gbasesit/sec. Ten nodes will consume the data with the rate of 10Gbasesit/sec, which is the limiting speed for the standard 10Gbasesit Ethernet cable connecting the cluster's rack with the switch. The nodes are being allocated on the same rack, which is the default SLURM behaviour.

Scalability can be improved by overriding the default behaviour of SLURM and allocating the nodes not from the same rack, but randomly from all available racks ("HPC random", open squares in Figure 4). Allocating the nodes on random racks allows one to engage more nodes without network saturation. For our cluster we could go up to 30-35 nodes with perfect linear scaling. For the most resources used (58 nodes) the deviation from a linear speedup is $\approx 7\%$ i.e. 5.50 minutes against the ideal 5.14, see Table 4. The threshold number of nodes in this strategy ($\approx 35$) is because of the uplink cable with the throughput of 50Gbasesit/sec being saturated. The proposed HPC strategies aimed at getting the maximum performance from the storage resources show that while even properly adjusted and tuned, the HPC approaches suffer from the network saturation at higher number of nodes. HDFS on the other hand maintains data locality, reducing communication and data transfer and hence lead to better scalability. Our Hadoop cluster has no more free nodes to continue to investigate the scaling as in plot at Figure 4, but we do not expect any significant deviations, since the observed behaviour is a generic for Hadoop with HDFS [15, 16].

## Usability aspects

A popular way to construct and execute bioinformatic pipelines on HPC resources is via the Galaxy [25] platform, which provides a Web-based graphical user interface (GUI) to bioinformatic programs, simplifying the experience for the end user. An alternative for Hadoop is Cloudgene [26], which is a light-weight and flexible Web-based solution for both public and private clouds. We implemented our Hadoop-pipeline in Cloudgene and extended the platform with functions to import data from the central file system at UPPMAX into HDFS on the private cloud (Figure 3). For our particular task in DNA sequencing, Cloudgene makes it easy to select data and parameters for its execution. Most of the data managing work is done automatically and the results can be downloaded to the client machine. The modular structure allows modification of the source code to adapt to the existing computing center architecture. For example, UPPMAX users can import their data from the sequencing platform directly to the Hadoop cluster by pressing a button and entering the credentials, being at the same time sure that their sensitive data is kept privately.

## Discussion

In this report we have described two approaches for high-performance analysis of DNA sequencing data; one based on regular HPC using a batch system and one based on Hadoop. We show that Hadoop installed on a private cloud is an appealing solution both on terms of runtime performance and also in terms of usability. A key result is the data size where Hadoop is favorable to regular HPC batch systems, and in order to compare them we developed highly optimized pipelines for both scenarios. We found that dataset sizes larger than 100 Gbases is where Hadoop excels over HPC, and also that Hadoop shows almost linear scalability for this problem. Extrapolation to infinite dataset size reveals however that HPC provides the results faster, given the same amount of resources as Hadoop on a private cloud.

We modified the preprocessing stage of the Crossbow software to enable it to make use of all nodes in the Hadoop system, end used a highly optimized Bash script to compare the scaling relations between the ratio of the alignment time to the communication time as a function of the reciprocal number of nodes on an HPC system. Our results show that the calculations on Hadoop with HDFS scales better than the network attached parallel storage commonly used in the HPC centers. In addition we show how performance of the HPC approach can be improved by redefining the queueing system's default behaviour. This is however not of practical use in an HPC center since the network bandwidth is shared among multiple users. Finally, we demonstrate that an existing and extensible publically available web-based GUI (Cloudgene) provides an easy way to execute bioinformatics analysis on Hadoop for those who are less experienced with Linux scripting.

## Methods

### Datasets

We used publicly available DNA-seq datasets (I–III) as well as a synthetic dataset (IV) of *A.thaliana*, the well-known model plant, and a dataset (V) of two *H.sapiens* individuals (Table 5). Data for datasets I-III and V were obtained using Illumina/HiSeq sequencing platforms. Further information about the datasets is provided in the Supplementary material section.

### Analysis pipelines

We constructed two pipelines for identifying single-nucleotide polymorphisms (SNPs) from short read data; one for Hadoop and one for HPC. Acknowledging that there are different approaches and software for conducting bioinformatic analysis for HPC (e.g., GATK [27]), we decided to create the analysis pipeline as simple as possible to be able to make a better comparison with the one implemented for Hadoop.

1. HPC approach

   Short read alignment: Bowtie ver. 0.12.8

   SNP calling: Samtools ver. 0.1.19

2. Hadoop approach: Crossbow

    Short read alignment: Bowtie ver. 0.12.8

    SNP calling: SOAPsnp 1.02 [28]

In the HPC pipeline, reads were aligned with Bowtie, followed by sorting the mapped reads and SNP calling with Samtools. The Bowtie aligner natively implements OMP, meaning that with 8 cores on the same computer the result can be theoretically obtained 8 times faster than on a single core. Likewise, Samtools (as of version 0.1.19) also offers shared memory parallelism for several of its functions. Where available these features were used to improve the analysis speed. The exact workflow used is available from github [24].

The equivalent Hadoop-based pipeline was implemented with Crossbow. The input data and the indexed genome reference were copied to Hadoop's storage system (HDFS) before starting the experiments. Crossbow implements a short pipeline that pre-processes the input data, transforming it into a format suitable for the alignment stage, and then continues to use Bowtie for alignment and SoapSNP to call SNPs. Unfortunately, Crossbow's preprocessor is not written in MR manner, and thus cannot be run in a massively parallel way. Due to this limitation, this basic step threatened to be the most time-consuming procedure in our test pipeline and bias our experiments. To overcome this bottleneck we substituted Crossbow's preprocessor with our own MR implementation (available from github [24]).

### Computational resources

To run the HPC analysis pipeline we used a computational cluster at UPPMAX, which is heavily used for analysis of NGS-data [29], on nodes equipped with 8 dual-core CPUs. Data and reference genomes were stored on a parallel, shared-storage system. The Hadoop test platform was deployed on a private cloud at UPPMAX using the OpenNebula [30] cloud manager. The cluster was set up with Cloudera Hadoop distribution version 2.0.0-mr1-cdh4.5.0 [31]. For details on computational resources, see Supplementary material.

## Acknowledgments

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. We thank system experts Pontus Freyhult and Peter Ankerstål at UPPMAX for valuable discussions on effective storage and network usage, Jonas Hagberg (BILS, Stockholm, Sweden) for implementing the Cloudgene extensions to import data from UPPMAX filesystem, as well as Wesley Schaal for proofreading.

## References

1. Metzker ML (2010) Sequencing technologies – the next generation. Nat Rev Genet 11: 31–46.

2. Marx V (2013) Biology: The big challenges of big data. Nature 498: 255-60.

3. Hiseq Comparison. URL `http://www.illumina.com/systems/sequencing.ilmn`.

4. Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25: 1754–1760.

5. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10: R25.

6. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. (2009) The Sequence Alignment/Map format and SAMtools. Bioinformatics 25: 2078–2079.

7. The OpenMP® API specification for parallel programming. http://openmp.org/. URL http://openmp.org/.

8. List statistics. URL http://www.top500.org/statistics/list/.

9. Tange O (2011) Gnu parallel - the command-line power tool. ;login: The USENIX Magazine 36: 42-47.

10. The Message Passing Interface (MPI) standard. http://www.mcs.anl.gov/research/projects/mpi/. URL http://www.mcs.anl.gov/research/projects/mpi/.

11. pMap: Parallel Sequence Mapping Tool. http://bmi.osu.edu/hpc/software/pmap/pmap.html. URL http://bmi.osu.edu/hpc/software/pmap/pmap.html.

12. The Extended Randomized Numerical alignEr. http://erne.sourceforge.net.

13. Clement NL, Snell Q, Clement MJ, Hollenhorst PC, Purwar J, et al. (2010) The GNUMAP algorithm: unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. Bioinformatics 26: 38-45.

14. Dean J, Ghemawat S (2004) MapReduce: Simplified data processing on large clusters. Sixth Symposium on Operating System Design and Implementation: 2004; San Francisco, CA .

15. Lin J, Dyer C (2010) Data-Intensive Text Processing with MapReduce. Morgan and Claypool Publishers.

16. White T (2009) Hadoop: The Definitive Guide. O'Reilly, first edition edition. URL http://oreilly.com/catalog/9780596521981.

17. Sammer E (2012) Hadoop Operations. O'Reilly Media, Inc., 1st edition.

18. Langmead B, Schatz MC, Lin J, Pop M, Salzberg SL (2009) Searching for SNPs with cloud computing. Genome Biol 10: R134.

19. The NCBI Sequence Read Archive. URL http://www.ncbi.nlm.nih.gov/Traces/sra.

20. Schneeberger K, Ossowski S, Lanz C, Juul T, Petersen AH, et al. (2009) SHOREmap: simultaneous mapping and mutation identification by deep sequencing. Nat Meth 6: 550–551.

21. Biostars forum. URL http://www.biostars.org/p/48781.

22. Pireddu L, Leo S, Zanetti G (2011) SEAL: a distributed short read mapping and duplicate removal tool. Bioinformatics 27: 2159-60.

23. Bozdag D, Hatem A, Catalyurek UV (2010) Exploring parallelism in short sequence mapping using Burrows-Wheeler Transform. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on : 1–8.

24. Manuscript github repository. URL https://github.com/raalesir/mr_scripts.

25. Giardine B, Riemer C, Hardison RC, Burhans R, Elnitski L, et al. (2005) Galaxy: a platform for interactive large-scale genome analysis. Genome Res 15: 1451-5.

26. Schonherr S, Forer L, Weissensteiner H, Kronenberg F, Specht G, et al. (2012) Cloudgene: a graphical execution platform for MapReduce programs on private and public clouds. BMC Bioinformatics 13: 200.

27. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, et al. (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 20: 1297-303.

28. Short Oligonucleotide Analysis Package. http://soap.genomics.org.cn/soapsnp.html.

29. Lampa S, Dahlö M, Olason P, Hagberg J, Spjuth O (2013) Lessons learned from implementing a national infrastructure in Sweden for storage and analysis of next-generation sequencing data. GigaScience 2: 9.

30. Open Nebula. http://opennebula.org.

31. Cloudera. http://www.cloudera.com/content/cloudera/en/why-cloudera/hadoop-and-big-data.html.

32. Milou Cluster. http://www.uppmax.uu.se/the-milou-cluster.

33. Gulo Storage. http://www.uppmax.uu.se/gulo.

## Supplementary material

### Datasets

The datasets used in the paper are publicly available at:

I: `http://1001genomes.org/data/software/shoremap/shoremap_2.0\/data/reads/Schneeberger.2009/Schneeberger.2009.single_end.gz`

II: `http://1001genomes.org/data/software/shoremap/shoremap_2.0/data/reads/Galvao.2012/Galvao.2012.reads1.fq.gz,http://1001genomes.org/data/software/shoremap/shoremap_2.0/data/reads/Galvao.2012/Galvao.2012.reads2.fq.gz`

III: `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`

IV: artificial pair-ended dataset for *A.thaliana* created with the `wgsim` program from the Samtools package.

V: `http://www.ncbi.nlm.nih.gov/sra/SRX148888`

### Reference genomes

- TAIR10 for datasets II-IV `ftp://ftp.arabidopsis.org/home/tair/Sequences/whole_chromosomes/*.fas`
- TAIR8 for dataset I `ftp://ftp.arabidopsis.org/home/tair/Genes/TAIR8_genome_release/`
- H.sapiens, NCBI v37 `ftp://ftp.ccb.jhu.edu/pub/data/bowtie_indexes/h_sapiens_37_asm.ebwt.zip`

### Description of computational facilities

1. HPC: Multinode short read alignment was performed on the Milou cluster [32], equipped with dual 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 10Gbasesit/s uplink.

2. Storage: Gulo [33] is a custom built Lustre 2.4 system using 8 HP nodes with MDS600 storage boxes and an additional node for metadata handling. In total, it provides roughly 1 PB of storage and is accessed with Lustre's own protocol. It supports data striping over multiple nodes and disk targets and can give a theoretical single file read performance of up to 80 Gbasesit per second.

3. Our Hadoop test platform was deployed on a private cloud at UPPMAX using the OpenNebula [30] cloud management system. Each node in this deployment was equipped with dual 4-core Intel Xeon 5420 (2.50 GHz; 12 MB L2 cache), 16 GB RAM, one 1 TB SATA disk and Gigabit Ethernet. The cluster was set up with Cloudera Hadoop Distribution version 2.0.0-cdh4.4.0 [31]. Note that the physical hardware provided less RAM than desired. Each VM node has 7 cores where each can use less than 2 GB of RAM, which is little for Hadoop. We expect much better performance with twice as much memory.

### Bash script for the preprocessing stage

```
pbzip2 -dc $file1 | paste - - - - -d'\t' | cut -f1,2,4 | paste - -d' ' <(pbzip2 -dc $file2
| paste - - - - -d'\t' | cut -f2,4) | pbzip2 -cz > $fileOut
```

Journal abbreviations in references are inconsistant. The journal name for ref 2 is wrong. Capitalization of refs 4, 14, 16, 18, 19, 25, 27, etc are wrong. All refs should be checked for accuracy and style consistancy. Many or all footnotes should be changed to endnotes (references).

## Figure Legends

## Tables

**Table 1.** Timings (in minutes) for the sort routine of the Samtools package for the dataset IV (58 Gbases BAM file). The measurements are done on a single HPC node with $p = 16$ cores. Averaging is done over 5 independent simulations. The amount of RAM is given for all 16 cores.

| RAM, GB | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| timing, minutes | $208 \pm 50$ | $172 \pm 13$ | $136 \pm 12$ | $87 \pm 11$ |

Figure 1

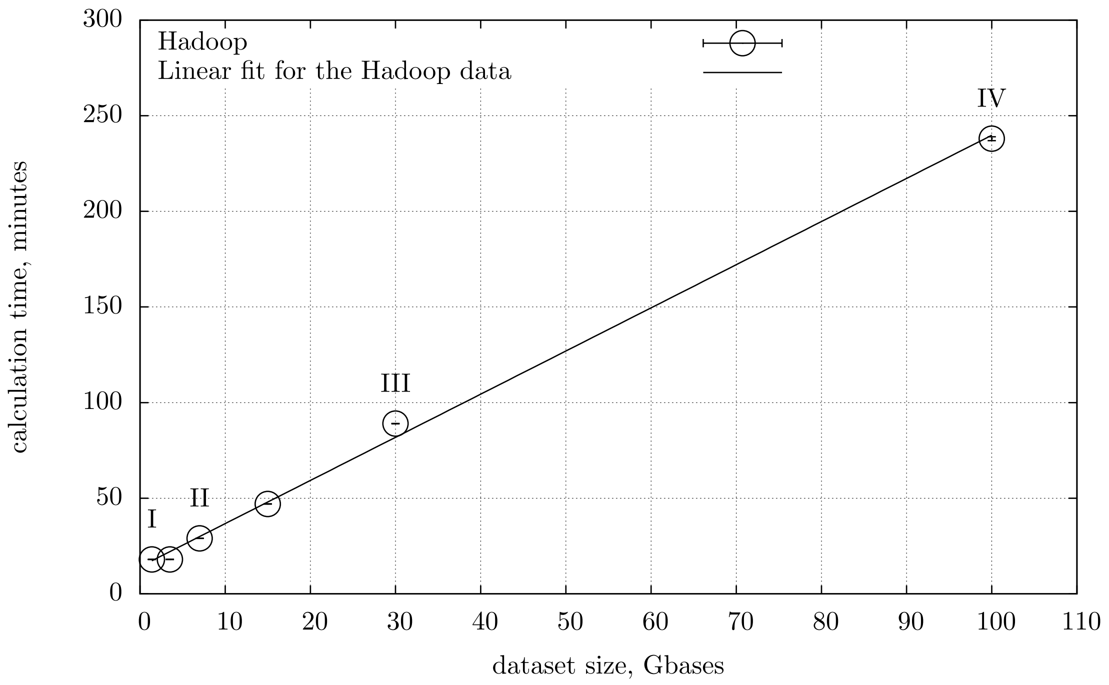


**Figure 1.** Calculation time for selected datasets (labeled with roman numerals) for the 56 cores Hadoop cluster shows a linear scaling with dataset size.

**Table 2.** Timings (in minutes) with Crossbow's native read preprocessor and with derived approach for datasets II, III, IV on a 56 cores Hadoop cluster, using the Crossbow, and with the BASH script on a single HPC node with 16 cores.

| Dataset | II (7 Gbases ) | III (30 Gbases) | IV (100 Gbases) |
|---|---|---|---|
| Crossbow native | $60.6 \pm 0.7$ | $299.0 \pm 2.3$ | $673 \pm 1.0$ |
| Hadoop, this work | $7.0 \pm 0.0$ | $20.7 \pm 0.4$ | $52.4 \pm 0.1$ |
| BASH, this work | $7.4 \pm 0.0$ | $31.1 \pm 0.1$ | $114.5 \pm 0.3$ |

Figure 2

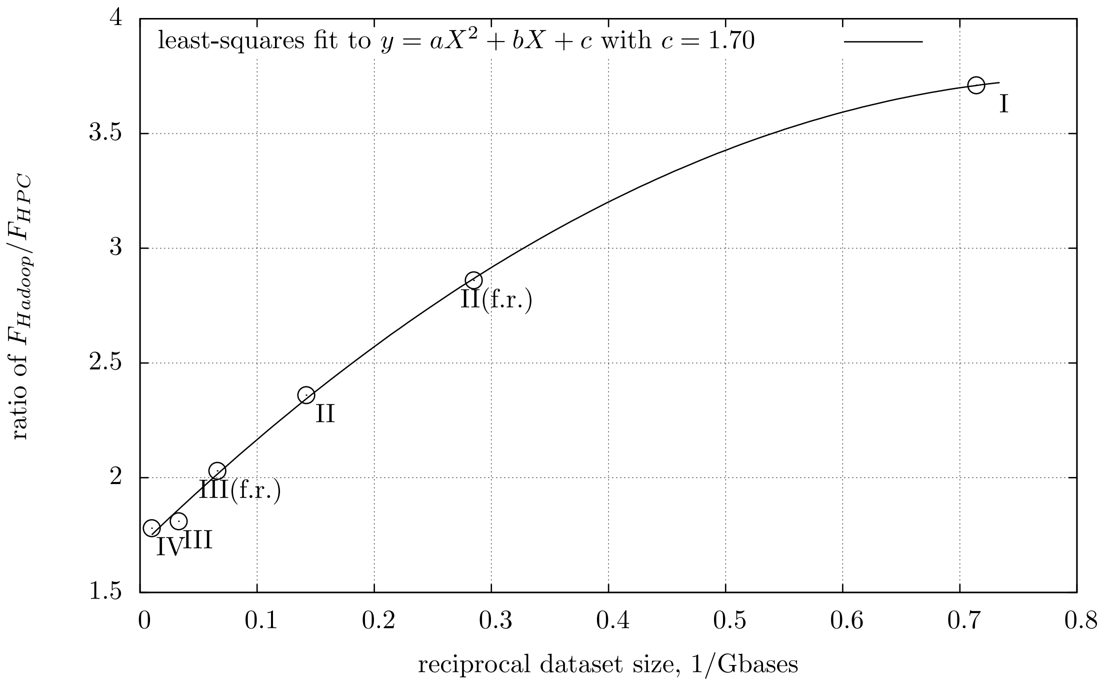
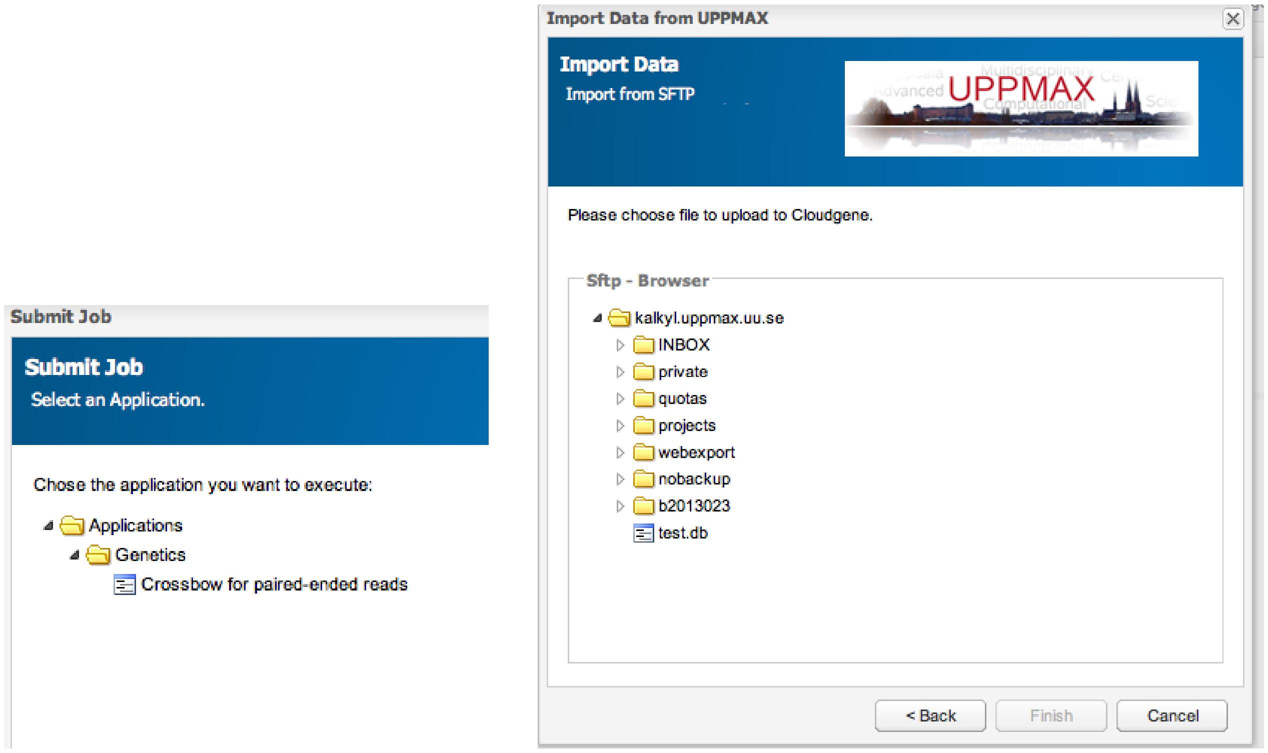


**Figure 2.** The ratio of the $F_{Hadoop}/F_{HPC}$ as a function of a reciprocal dataset size in Gigabases. Calculations were carried out for 56 cores Hadoop and 16 cores HPC cluster correspondingly. The points are fit to a quadratic least-squares curve, which makes it possible to predict *infinite* dataset size. Datasets are labeled with roman numerals.

**Table 3.** Timings for Dataset I processed on different number of cores of a HPC node. awkward title, consider replacing with graph (and new title) better?

| $N$ cores | timing, minutes | | | speed-ups | | |
|---|---|---|---|---|---|---|
| | alignment | SNP calling | total | total | map | SNP call |
| 1 | 22.75 ±1.24 | 26.1±0.7 | 48.9 | 1.00 | 1.00 | 1.00 |
| 2 | 11.00±0.05 | 16.9±1.3 | 27.9 | 0.88 | 1.03 | 0.77 |
| 4 | 5.91±0.04 | 11.7±0.5 | 17.6 | 0.69 | 0.96 | 0.56 |
| 6 | 4.24±0.04 | 8.6±0.8 | 12.8 | 0.64 | 0.89 | 0.51 |
| 8 | 3.44±0.03 | 8.6±0.9 | 12.1 | 0.51 | 0.83 | 0.38 |
| 10 | 2.91±0.01 | 7.7±0.5 | 10.6 | 0.46 | 0.78 | 0.34 |
| 12 | 2.64±0.04 | 7.5±0.5 | 10.1 | 0.40 | 0.72 | 0.29 |
| 14 | 2.57±0.03 | 7.5±0.6 | 10.1 | 0.35 | 0.63 | 0.25 |
| 16 | 2.88±0.06 | 7.4±0.4 | 10.2 | 0.30 | 0.49 | 0.22 |


Submit Job

Submit Job

Select an Application.

Chose the application you want to execute:

Applications

Genetics

Crossbow for paired-ended reads

Import Data from UPPMAX

Import Data

Import from SFTP

Please choose file to upload to Cloudgene.

Sftp - Browser

kalkyl.uppmax.uu.se

INBOX

private

quotas

projects

webexport

nobackup

b2013023

test.db

(a) (b)

**Figure 3.** An example of a job setup with the graphical Hadoop front-end Cloudgene, providing a smooth user experience even for novice users. **a)** Pipeline selection - in our case containing the Crossbow pipeline. **b)** The UPPMAX-adapted functionality to browsing and import data from the user's home folder in the shared file system.

**Table 4.** Timings (in minutes) for HPC and Hadoop deployments for different dataset sizes. Dataset is shown in brackets by roman numerals, "f.r." stands for "forward reads". Big deviation what aspect of deviation, not fixable? for the HPC is due to Samtools BAM sorting, which is IO and memory intensive, thus strongly depend on the queuing at the HPC cluster. No p= used here.

| data, Gbases | 1.4<br>(I) | 3.5<br>(II, f.r.) | 7.0<br>(II) | 15.0<br>(III, f.r.) | 30.0<br>(III) | 100.0<br>(IV) | 250<br>(V) |
|---|---|---|---|---|---|---|---|
| Hadoop, 56 cores | – | – | 39 | 62 | 108 | 250 | 1125 |
| Hadoop, 56 cores | $18 \pm 0$ | $18 \pm 0$ | $29 \pm 0$ | $47 \pm 0$ | $89 \pm 0$ | $238 \pm 1$ | $1164 \pm 14$ |
| HPC1, 8 cores | 41 | 96 | 157 | 307 | 596 | 1490 | – |
| HPC, 16 cores | $17 \pm 1$ | $22 \pm 6$ | $43 \pm 3$ | $81 \pm 9$ | $172 \pm 15$ | $467 \pm 60$ | $> 48$ hours |

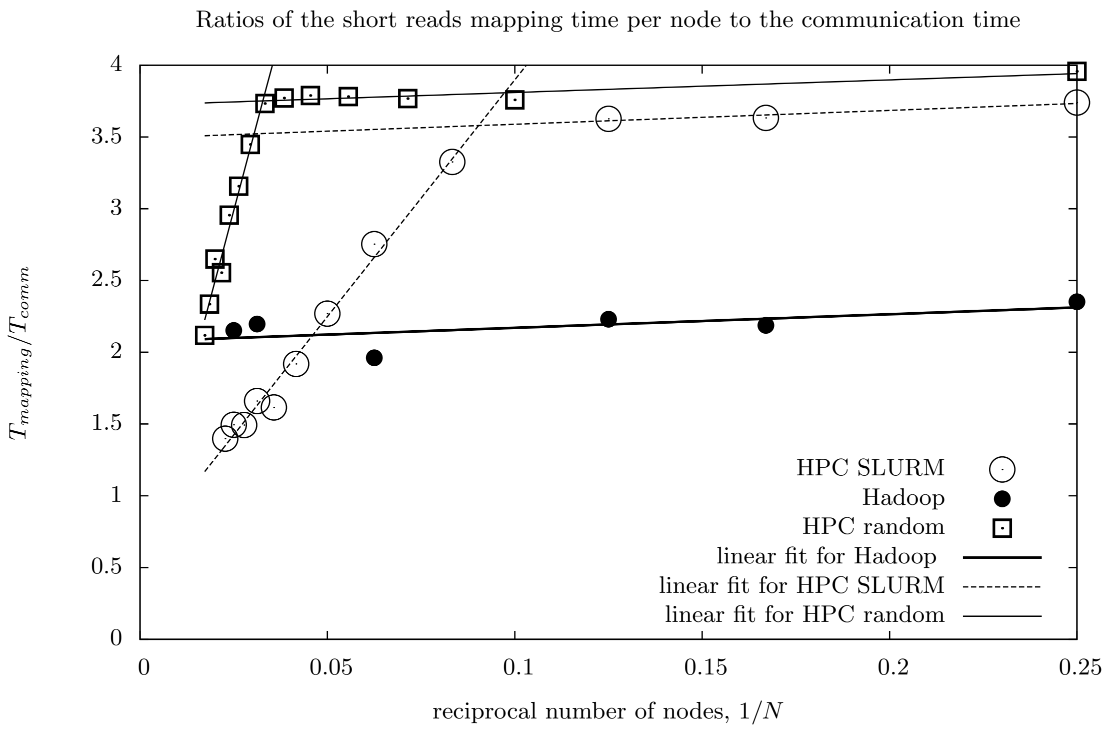


**Figure 4.** Ratios of the alignment time $T_{alignment}$ to the communication costs $T_{comm}$ for HPC and Hadoop clusters for Dataset IV as a function of reciprocal number of nodes $1/N$. Two HPC scenarios are shown as "HPC SLURM" and "HPC random", which correspond to standard SLURM behaviour and a modified one where nodes are being allocated from random racks. Linear fit to $ax + b$ was done with the least-squares method. The estimated values for Hadoop approach are $(a, b) \approx (0.95, 2.07)$. One can see two defined linear regions for the HPC approach with very different tangents for linear and non-linear scalings correspondingly. For the "HPC SLURM" $(a, b) \approx (0.96, 3.49)$ and $(a, b) \approx (32.99, 0.60)$, and for the "HPC random" $(a, b) \approx (0.88, 3.72)$ and $(a, b) \approx (98.24, 0.53)$. The constant term for HPC goes below unity, i.e. for the large number of nodes the communication will take more time than actual alignment.

**Table 5.** Datasets used in the comparison.

| dataset | organism | size in Gbases |
|---|---|---|
| I | *A.thaliana* | 1.4 |
| II | *A.thaliana* | 7.0 |
| III | *A.thaliana* | 30.0 |
| IV | *A.thaliana*, the artificial dataset created using Samtools package | 100.0 |
| V | *H.sapiens*, two individuals (GM12750 and GM12004), sample SRR499924 | 250.0 |

**Table 6.** Timings for alignment and the ratio $T_{alignment}/T_{comm}$ for HPC and Hadoop clusters for Dataset IV. For the "HPC random" approach, data chunks have to be copied to the local scratch disks first and the alignments (SAM files) copied back while Hadoop keeps all the data inside HDFS and hence does not need data staging. Hadoop however needs to preprocess reads before the actual alignment stage in order to be able to operate in MR manner resulting in what we term "communication costs". Note that each HPC node has 16 cores, while each Hadoop node has 7 (one core is dedicated to run the virtual machine).

| Hadoop | | | HPC random | | |
|---|---|---|---|---|---|
| Number of nodes (cores) | Mapping time, minutes | $\frac{T_{alignment}}{T_{comm}}$ | Number of nodes (cores) | Mapping time, minutes | $\frac{T_{alignment}}{T_{comm}}$ |
| 4(28) | 293.5 | 2.33 | 4(64) | 74.4 | 3.89 |
| 6(42) | 189.8 | 2.19 | 10(160) | 32.4 | 3.76 |
| 8(56) | 136.0 | 2.23 | 14(224) | 22.7 | 3.77 |
| 16(112) | 70.3 | 1.96 | 18(288) | 17.9 | 3.78 |
| 32(224) | 39.3 | 2.20 | 22(352) | 14.5 | 3.79 |
| 40(280) | 32.5 | 2.15 | 26(416) | 12.3 | 3.77 |
| | | | 30(480) | 10.7 | 3.73 |
| | | | 34(544) | 9.5 | 3.45 |
| | | | 38(608) | 8.5 | 3.16 |
| | | | 42(672) | 7.6 | 2.96 |
| | | | 46(736) | 7.0 | 2.55 |
| | | | 50(800) | 6.4 | 2.65 |
| | | | 54(864) | 5.9 | 2.34 |
| | | | 58(928) | 5.5 | 2.12 |