A quantitative assessment of the Hadoop framework for analysing massively parallel DNA sequencing data (Hadoop Or Not Hadoop)

Alexey Siretskiy, Luca Pireddu, Ola Spjuth [specify affiliation]

February 19, 2014

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 analyse 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 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 observered 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. Brit spelling assumed since that's most common in this document

1 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. As an example of the scale of the demands, consider that a single Illumina high-throughput sequencing run produces approximately 3 TB of raw data in 10 days [3]. Indeed, the Swedish UPPMAX[1] [unusual to use both footnotes and endnotes] high-performance computing (HPC) center recently disclosed data showing that just in their sequencing

[1] uppmax.uu.se

context (most sequencing performed in Sweden) huh? storage is being occupied at a rate of 1 TB/day while the analyses are using over 1 million computing core-hours per month [4].

A common step of NGS data analysis consists of mapping short reads to a reference sequence and then finding the genetic variations specific to the sample. ~~Most of the bioinformatic programs are written for the Linux operating system.~~ /COMMENTawkward but maybe it's needed Some of the most widespread tools[2] like BWA [5], Bowtie [6] and Samtools [7] are "regular" computer programs, not made with distributed computing in mind. Many others do not even have the native ability to use multiple cores ~~on the same computer~~[3].

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

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 ~~by a programmer~~ and there are only a few existing solutions available for processing sequencing data [11, 12, 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 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]. Fix bib error (can't have both editor and author for this ref type) 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

[2] average number of invocations per month during 2013 at UPPMAX: Samtools – 30000, BWA – 25000, Bowtie – 5000; from internal communications

[3] Samtools acquired this feature in v.0.1.19

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 *if* and *when* that's two questions 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 use five datasets with short reads of different sizes, align them against the appropriate reference genome, and call the variances. 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. The execution times for each dataset were collected and scaling relations were analysed to answer the questions in focus.

The manuscript is structured as follows: [odd to do this in a normal publication, but OK] In Section II we briefly introduce the datasets, computational facilities, analysis pipelines design, and the software used. Then, Section III presents experimental results; first verifying that both HPC and Hadoop approaches extract the same mutation and then investigating the scaling relations in terms of data size and computing resources. The results are discussed in Section IV, and conclusions in Section V. Supplementary materials are provided in the corresponding section.

2 Methods

2.1 Datasets

We used publicly available DNA-seq datasets (I–III), a synthetic dataset (IV) of *A.thaliana*, the well-known model plant, and dataset (V) of two *H.sapiens* individuals (Table 1). 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.

Table 1: Datasets used

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

2.2 Analysis pipelines

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). Our experiments then consisted of running the pipelines for the selected input dataset and measuring the wall-clock run time for each pipeline stage. All experiments were repeated several times and the results averaged considering the variations in table 4, why not use medians to obtain a data point for the particular combination of data size and computational platform. Acknowledging that there are different approaches and software for conducting bioinformatic analysis for HPC (e.g., GATK [18]), we decided to create the analysis pipeline as simple as possible to be able to pass huh? the same stages on Hadoop and HPC:

1. HPC approach
   Short read alignment: Bowtie ver. 0.12.8
   SNP calling: Samtool ver. 0.1.19
2. Hadoop approach: Crossbow [19]
   Short read alignment: Bowtie ver. 0.12.8
   SNP calling: SOAPsnp 1.02 [20]

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 in the code repository created for this work [**?**]. missing ref

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.

2.3 Computational resources

To run the HPC analysis pipeline we used a computational cluster at UPPMAX on nodes equipped with 8 dual-core CPUs. Data and reference genomes were stored on a parallel, shared-storage system[21]. The Hadoop test platform was deployed on a private cloud at UPPMAX using the OpenNebula [22] cloud manager. The cluster was set up with Cloudera Hadoop distribution version 2.0.0-mr1-cdh4.5.0 [23]. For details on computational resources, see Supplementary material.

3 Results

3.1 Modified preprocessing stage

A native preprocessing stage in Crossbow[4] provides great flexibility in delivering the data to the Hadoop cluster. Data can be downloaded from Amazon S3, FTP and HTTP servers over the Internet[19]. The only way to introduce parallelisations 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.

Being pragmatic we assume that sequencing platforms are usually affiliated with computation facilities which provide both storage and computers, therefore we considered that the data already been downloaded to storage Hadoop could access. We rewrote the preprocessing stage scripts in a MR-manner, benefiting from its massively parallel nature. The requirement is that the FASTQ data are BZIP2 archived 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 ingested special word? to the HDFS. BZIP2 provides very good compression and is splittable, meaning that the archive can be expanded in a parallel manner[5]. Also, BZIP2 format is natively supported by Hadoop by enabling the respective codec, i.e., ~~for the developer~~ there is no difference in dealing with the FASTQ data or with its BZIP2 archive.

The Hadoop streaming library offers a possibility to write Mapper and Reducer 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).ref? 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 the HDFS beforehand, to eliminate possible network delays, e.g., while downloading data from some Amazon cloud. Table 2 illustrates the benefits of our approach. To complete the comparison the same preprocessing was performed with

Table 2: Timings (in minutes) with the Crossbow native read preprocessor and with derived approach for datasets II, III, IV on ~~$p = 56$~~ 56 cores Hadoop cluster, using the Crossbow, and with the BASH script on a single HPC node ~~, $p = 16$~~ with 16 cores. A bit odd to use the vertical lines before/after table, but OK.

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

[4] the same holds for Myrna, a cloud-based solution for differential expression analysis

[5] other options like splittable LZO are also possible

the BASH scripting against the data located on the HPC storage. The script[6] ran on a single HPC node, utilizes multiple cores, with the limiting factor being the HDD IO performance, which effectively leaves with 5-7 cores of 16. awkward

3.2 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[24] was successfully localized by both applications.

3.3 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). Consider replacing with a graph. ~~As one can see~~ the aligning process with the Bowtie scales fairly well, but the SNP calling part is a bottleneck, scaling worse than the Bowtie, and consuming a progressively larger portion of the total calculation time. 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 of using Hadoop MR is its almost linear scalability, i.e., calculation time linearly depends on the size of the dataset[19, 25], regardless the scaling nature of the underlying program (Samtools, Bowtie, etc.). Figure 1 this figure should be closer, table and figure layout odd in general , based on Table 4, shows the calculation time as a function of the dataset size for $p = 56$ is this 'p=' notation necessary cores Hadoop cluster. Dataset V was excluded since it is for *H.sapiens*, which has more than 20 times larger genome than *A.thaliana*. so what To stress the linear nature of scaling, both sets of points were fit ~~to linear polynomial~~ using the least squares method.

3.4 Comparing Hadoop and HPC runtime efficiency for different dataset sizes

delete 'for different dataset sizes' 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 following function:

$$F = T_p \times p,$$

where T_p is the calculation time on p cores. This trivial/obvious formula doesn't deserve such special treatment. Using the data from Table 4, we plot the ratio

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

[7] We used special tricks to parallelize the Samtools analysis by chromosome, as exemplified here http://www.biostars.org/p/48781.

Figure 1

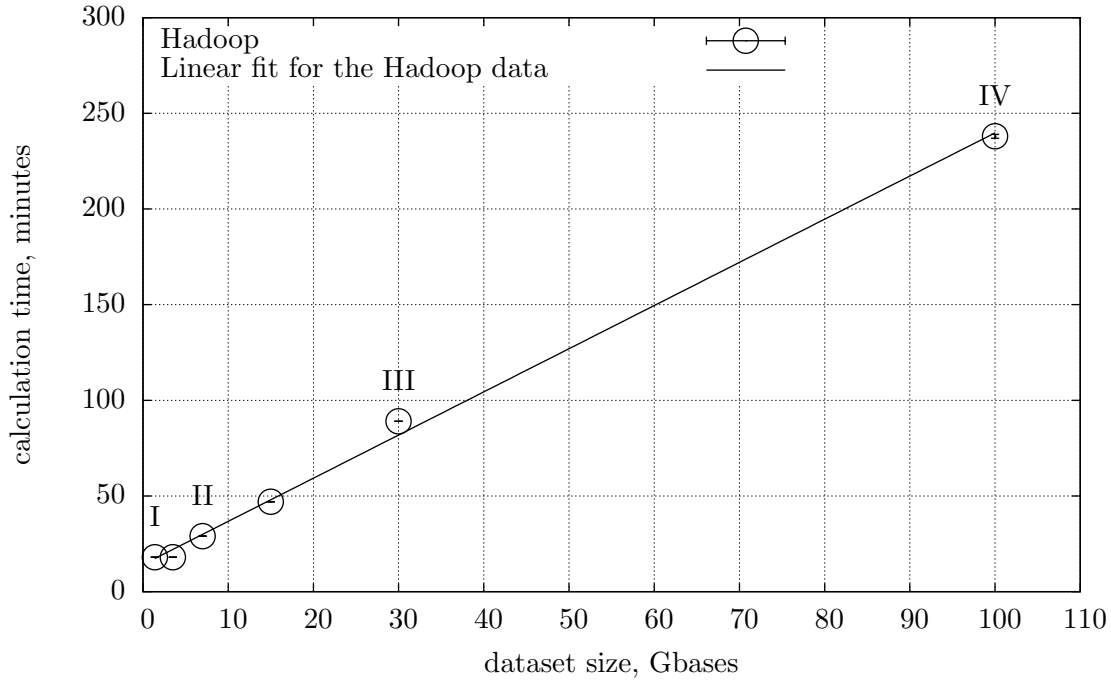


Figure 1: Calculation time depending on the size of selected datasets for $p = 56$ p= returns cores Hadoop cluster on the private Cloud. ~~The least squares fit reveals linear scaling.~~ Datasets are marked with roman numerals. Rather large graph for such a simple result.

Table 3: Calculation time for Dataset I executed on different number of cores on a node of HPC cluster. awkward title, consider replacing with graph (and new title)

N cores	timing, minutes			speed-ups		
	mapping	SNP calling[7]	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

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

F_{Hadoop}/F_{HPC} ~~, keeping in mind that the closer the ratio is to the unity, the closer Hadoop's efficiency is to that of the HPC approach~~.

The curve in Figure 2, based on Table 4~~, is plotted for $p = 56$ cores Hadoop cluster and an HPC node ($p = 16$), and~~ displays the ratio F_{Hadoop}/F_{HPC} as a function of the reciprocal dataset size. 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 the ratio is 1.70 ± 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 linear one. Considering a *linear* scaling for the Hadoop approach for those datasets, Figure 1, one can conclude that HPC approach scales worse than linearly[8].

[8]For the linear scaling the curve of $F = T_p \times p$ does not depend on the dataset size, i.e. being a constant; thus the ratio of two constants would result a constant, but we observe a dependecy.

Figure 2

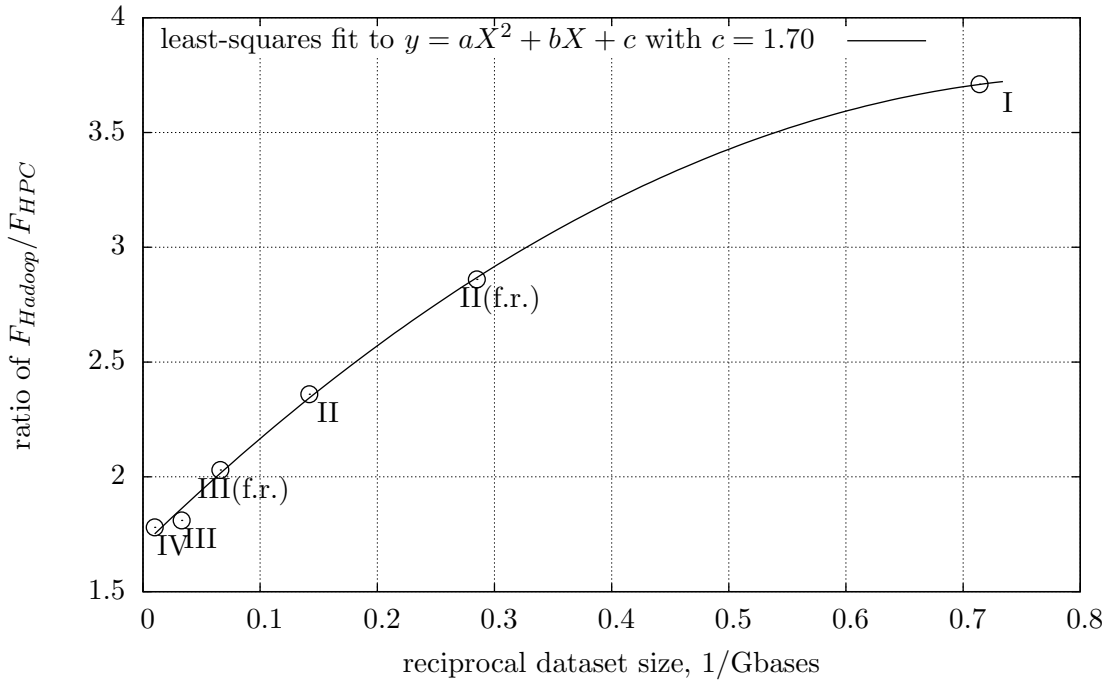


Figure 2: The ratio of the F_{Hadoop}/F_{HPC} as a function of a reciprocal dataset size in Gigabases. Calculations were carried out for $p = 56$ and $p = 16$ cores for Hadoop and HPC correspondingly. The points are fit to a quadratic least-squares curve, which makes it possible to predict *infinite*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

Why jump back and forth in the discussion of figures 1 and 2? We suspect the main reason for such a behavior is the fact that for large datasets (starting from dataset III in our case) the SNP calling routine with the Samtools becomes more memory and time demanding than the actual mapping with Bowtie. If lacking RAM, the Samtools 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 5 shows the role of the amount of allocated RAM on the Samtools sort timing, as a most time expensive huh? , for the dataset IV, starting from 8GB up to 80GB, performing 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 VM node.

Table 5: Timings (in minutes) for the sort routine of the Samtools package for the dataset IV (58Gb 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 ± 50	172 ± 13	136 ± 12	87 ± 11

3.5 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. We provide a comparison of HPC and Hadoop network communication costs depending on the number of nodes involved for a fixed dataset size (58 Gb, 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, $T_{mapping}$, 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]. Authors report almost linear scaling up to 32 nodes for pair-ended reads[9]. why footnote rather than endnote However, e.g., the Pmap package functions poorly ~~on UPPMAX cluster~~ for datasets larger than 20 Gbases ~~, raising~~

[9] http://bmi.osu.edu/hpc/slides/Bozdag10-HiCOMB.pdf, http://dna.cs.byu.edu/gnumap/HICOMB_Presentation.pdf

~~memory exceptions~~. Needs an intro like 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 [**?**], missing reference, written as 'repo' but assumed to be same as 'code_repo' above and compared the network performance with the standard Hadoop HDFS approach. We separated the mapping time, $T_{mapping}$, and the communication time, T_{comm}, and plotted $T_{mapping}/T_{comm}$ as a function of the reciprocal number of nodes $1/N$ (Figure 3). 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 mapping happens, and the sending of the aligned SAM[10] files back to the delivery location over the network.

Figure 3 shows the $T_{mapping}/T_{comm}$ ratio as a function of the reciprocal number of nodes $1/N$ for Hadoop and HPC approaches. why describe fig 3 again The HPC approach is presented in two versions that are based on a bit different strategies of the resource allocation.

Hadoop results (filled circles in Figure 3): One can see that the ratio $T_{mapping}/T_{comm}$ 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 mapping time and dataset chunk size D: $T_{mapping} \propto D \propto 1/N$, see [6], and Figure 1. Since the ratio $T_{mapping}/T_{comm}$ 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: The strategy named HPC SLURM[11] (open circles in Figure 3) is as follows: the data from the delivery location is split into chunks in parallel, which are simultaneously pushed to the local scratches 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[12] and unable to pass more data in a time unit, while the mapping time is still proportional to the chunk size: $T_{comm} \approx \text{const}, T_{mapping} \propto D \propto 1/N \rightarrow T_{mapping}/T_{comm} \propto 1/N$, i.e., a linear dependency, which one can observe in the plot. The transition area between the two modes – saturated and unsaturated – has the next origin huh? : each hard disk drive on the local node can write the data at a speed of about ~~100MB/sec~~$\approx$ 1Gbit/sec. Ten nodes will consume the data with the rate of 10Gbit/sec, which is the limiting speed for the standard 10Gbit 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

[10] http://samtools.sourceforge.net/SAMv1.pdf

[11] Simple Linux Utility for Resource Management

[12] The used storage at UPPMAX is a set of RAIDs (Redundant Array of Inexpensive Disks) with data striping, providing up to 80Gbit/sec of outcoming traffic 'outgoing' or 'incoming' or just 'traffic' .

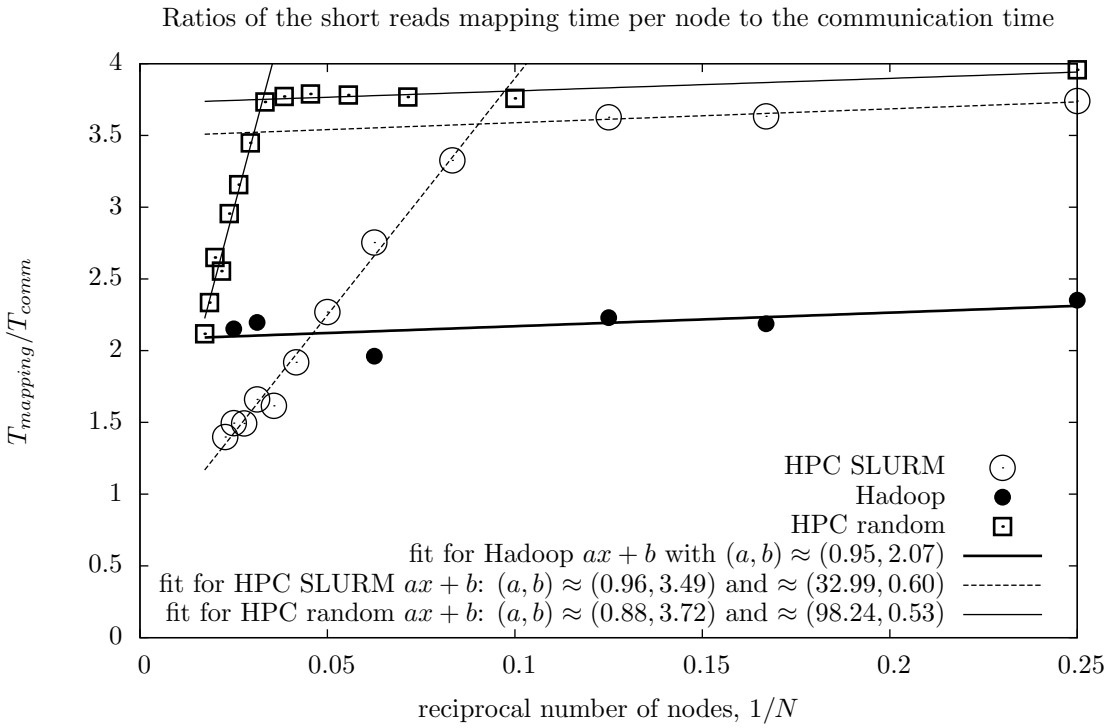


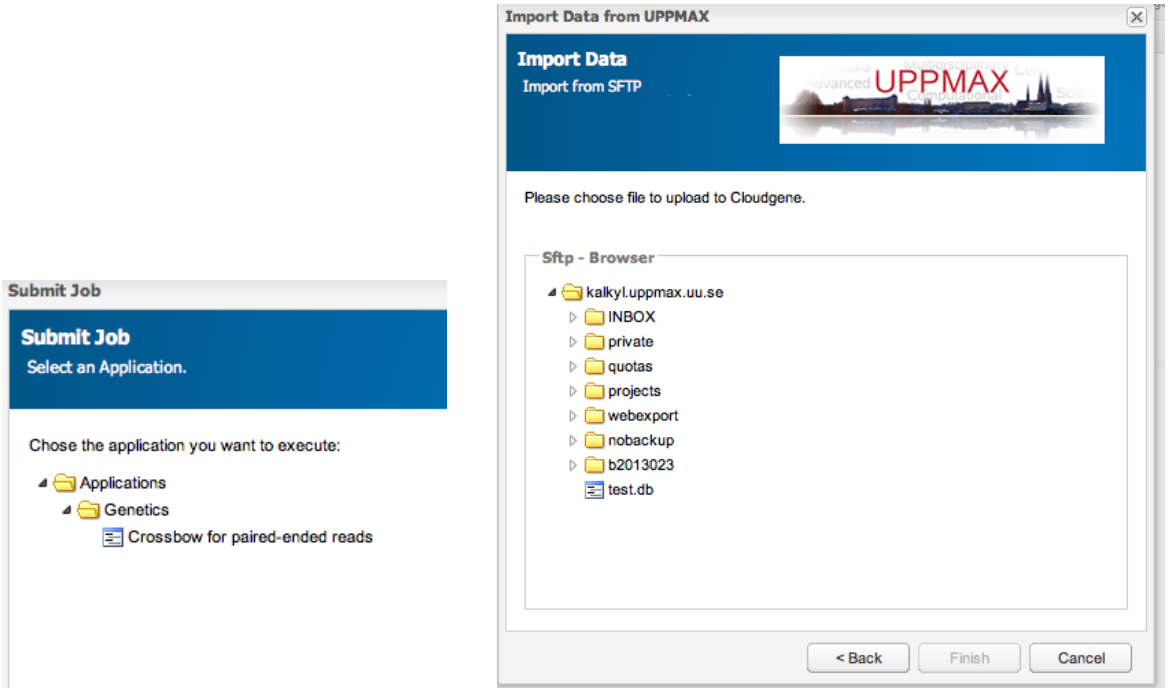
Figure 3: Ratios of the mapping time $T_{mapping}$ to the communication costs T_{comm} for HPC and Hadoop clusters for Dataset IV as a function of reciprocal cluster size $1/N$. Two HPC scenarios are shown as "HPC SLURM" and "HPC random", which correspond to standard SLURM behaviour and a modified one where nodes are being allocated from random racks. Linear fit done with the least-squares method. Figure legend too complex. Y-axis label inconsistant with previous figure.

allocating the nodes not from the same rack, but randomly from all available racks ("HPC random", open squares in Figure 3). 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 for the data. The threshold number of nodes in this strategy (≈ 35) is because of the uplink cable with the throughput of 50Gbit/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.

At the same time, the HDFS keeps data locality 'keeps data locally' or 'maintains data locality' , aiming to reduce the amount of communications, resulting less data move and, therefore, better scalability. Our Hadoop-in-the-Cloud cluster has no more (≈ 40) how does 40 approximate zero? free nodes to continue to investigate the scaling as in plot at Figure 3, but we do not expect any significant deviations, since the observed behaviour is a generic for Hadoop with HDFS. need ref?

Table 6: Timings for mapping and the ratio $T_{mapping}/T_{comm}$ 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	$\frac{T_{mapping}}{T_{comm}}$	Number of nodes (cores)	Mapping time, minutes	$\frac{T_{mapping}}{T_{comm}}$
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



(a) **(b)**

Figure 4: 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.

3.6 Usability aspects

A popular way to construct and execute bioinformatic pipelines on HPC resources is via the Galaxy[26] platform, which provides a Web-based graphical user interface (GUI) to bioinformatic programs, simplifying the experience for the end user. One of the alternatives for Hadoop is Cloudgene[27], 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 4(a)). For our particular task in DNA sequencing, Cloudgene provides the intuitive interface making one easy to follow. 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 centers architecture, Figure 4(b). 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 will be held locally, reducing the amount of unnecessary risks.

4 Conclusions

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 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 (Figure 1). Extrapolation to infinite dataset size (Figure 2) reveals however that HPC provides the results faster, given the same amount of resources as Hadoop on a private cloud.

To increase the performance of the existing Hadoop software, the modification to the preprocessing stage of the Crossbow/Myrna were suggested. The calculation time reduction can be observed in the Table 2. So, is Hadoop better after the modifications for infinite datasets?

Exploiting the embarrassingly parallel nature of the short reads mapping we used a highly optimized Bash script to compare the scaling relations between the ratio of the mapping time to the communication time as a function of the reciprocal number of nodes (Figure 3). 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. Should maybe mention somewhere that this sort of cheating. 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.

5 Acknowledgements

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 http://essenceofescience.se/home/. We thank system experts Pontus Freyhult and Peter Ankerstål at UPPMAX for valuable discussions on effective storage and network usage. We also thank Jonas Hagberg (BILS, Stockholm, Sweden), for implementing the Cloudgene extensions to import data from UPPMAX filesystem.

6 Supplementary material

6.1 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

6.2 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

6.3 Description of computational facilities

1. HPC: Multinode short read mapping was performed on the Milou cluster[28], 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 10Gbit/s uplink.
2. Storage: Gulo[21] 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 Gbit per second.
3. Our Hadoop test platform was deployed on a private cloud at UPPMAX using the OpenNebula [22] 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 [23]. Note that the physical hardware provided less RAM than desired. Each VM node has 7 cores, each of each can use less than 2 GB of RAM, which is little for Hadoop. We would expect much better performance with twice as much memory, as recommended.

   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).

References

[1] M L Metzker. Sequencing technologies – the next generation. *Nat Rev Genet*, 11(1):31–46, 2010.

[2] V Marx. Biology: The big challenges of big data. *Nature Technology Feature*, 498(7453):255–260, 2013.

[3] Hiseq comparison. http://www.illumina.com/systems/hiseq_comparison.ilmn.

[4] 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*, 2(1):9, 2013.

[5] H Li and R Durbin. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[6] B Langmead, C Trapnell, M Pop, and SL Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.

[7] H Li, B Handsaker, A Wysoker, T Fennell, J Ruan, N Homer, G Marth, G Abecasis, and R Durbin. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[8] The OpenMP® API specification for parallel programming. http://openmp.org/.

[9] Gnu parallel. http://www.gnu.org/software/parallel/.

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

[11] pmap: Parallel sequence mapping tool. http://bmi.osu.edu/hpc/software/pmap/pmap.html.

[12] The extended randomized numerical aligner. http://erne.sourceforge.net.

[13] N L Clement, Q Snell, M J Clement, P C Hollenhorst, J Purwar, B J Graves, B R Cairns, and W E Johnson. The gnumap algorithm: unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. *Bioinformatics*, 26(1):38–45, 2010.

[14] 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.

[15] Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers, 2010.

[16] Tom White. *Hadoop: The Definitive Guide*. O'Reilly, first edition edition, june 2009.

[17] Eric Sammer. *Hadoop Operations*. O'Reilly Media, Inc., 1st edition, 2012.

[18] A McKenna, M Hanna, E Banks, A Sivachenko, K Cibulskis, A Kernytsky, K Garimella, D Altshuler, S Gabriel, and M Daly. The genome analysis toolkit: A mapreduce framework for analyzing next-generation dna sequencing data. *Genome Research*, 2010.

[19] B Langmead, M C Schatz, J Lin, M Pop, and S L Salzberg. Searching for snps with cloud computing. *Genome Biology*, 10(11):R134, 2009.

[20] Short Oligonucleotide Analysis Package. http://soap.genomics.org.cn/soapsnp.html.

[21] Gulo storage. http://www.uppmax.uu.se/gulo.

[22] Open Nebula. http://opennebula.org.

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

[24] K Schneeberger, S Ossowski, C Lanz, T Juul, A H Petersen, K L Nielsen, J-E Jorgensen, D Weigel, and S U Andersen. Shoremap: simultaneous mapping and mutation identification by deep sequencing. *Nat Meth*, 6(8):550–551, 08 2009.

[25] L Pireddu, S Leo, and G Zanetti. Seal: a distributed short read mapping and duplicate removal tool. *Bioinformatics*, 2011.

[26] B Giardine, C Riemer, R C Hardison, R Burhans, L Elnitski, P Shah, Y Zhang, D Blankenberg, I Albert, J Taylor, W Miller, W J Kent, and A Nekrutenko. Galaxy: A platform for interactive large-scale genome analysis. *Genome Research*, 15(10):1451–1455, 2005.

[27] S. Schonherr, 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*, 13:200, 2012.

[28] Milou cluster. http://www.uppmax.uu.se/the-milou-cluster.