Hadoop Or Not Hadoop

Alexey Siretskiy, Luca Pireddu, Ola Spjuth

January 15, 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, emerged in the last decade, is a framework that automatically distributes data and computation and has been shown to scale to thousands of nodes. We here propose and report the quantitative comparison of Hadoop with regular high-performance computing resources for aligning short reads and calling variants for five datasets of different sizes up to 250 Gigabases. We modified and wrote new pieces of code in order to substantially increase the performance of existing software. Observing the scaling relations we are able to draw conclusions about the perspectives of the approaches. Our results show that as data set sizes reach 100 Gigabases, the Hadoop-based pipelines become performance-competitive with a canonical high-performance cluster solutions.

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 day [3]. Indeed, the Swedish UPPMAX[1] HPC center recently disclosed data showing that just in their sequencing context (most sequencing performed in Sweden) 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 (tools) are written for the Linux operating system. Some of

[1] uppmax.uu.se

the most widespread software tools[2], like BWA [5], Bowtie [6] and Samtools [7] are "regular" computer programs, not made up with distributed computing in mind; many others do not even have the native ability to use multiple cores on the same computer[3].

Most common way 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 usually does not exceed 16. For the tools which do not support the OMP natively, i.e. the Samtools, a variant calling can be parallelised specifying the option, creating a separate process for each chromosome, or using GNU Parallel [9] Linux utility. The fact of great importance, however, is that the multi-core approach does not improve the performance of operations that are limited by *disk* or *network* throughputs, motivating to split the dataset and use the multi-node parallelisation.

The 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 all the threads synchronisation (or load balancing) has to be coded by a programmer and there are only a few existing solutions available for processing sequencing data [11, 12, 13].

Another common way to introduce the parallelisation in Linux systems is to use Bash scripting e.g. involve already existing utilities and cluster tools, split the data into chunks to deliver and proceed them on the separate nodes, merging the results afterwards. This kind of solution benefiting from both MPI-like and OMP, is involved in our work as well. It provides great performance, but usually it is tightly binded to the local computational cluster and network architectures.

The Map-Reduce (MR) programming paradigm [14] offers a compelling alternative for running task in a *massively* parallel way. This paradigm, however, shifts the focus from the best performance to scalability, suited for managing huge datasets of size of terabytes [15]. The prevalent open source implementation of Map-Reduce is Hadoop [14, 16]. For the purpose of this work, we focus on the Cloudera Hadoop MR with the Hadoop Distributed File System (HDFS).

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]. HDFS is a complementary component that stores data by automatically distributing it over the entire cluster, writing data blocks onto each nodes local disks, therefore effectively moving the computation to the data and hence reducing network traffic. The 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 *if* and *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.

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

[3]Samtools acquired this feature since v.0.1.19

We use five datasets with short reads of different size to align them against the reference genome, and call the variances. For one dataset we made a check to identify the actual mutation to make sure that HPC and Hadoop give the same answers. From the algorithmic point of view we propose a modification of a code for a preprocessing stage for Crossbow in order to benefit fully from massively parallel nature of MR computations. For the classical HPC DNA-seq analysis programs we developed a general-use bash scripts, utilizing multiple nodes for short reads alignment and exploiting the network fully, speeding up the calculations. The execution times for each dataset were collected and scaling relations were analysed to answer the question in focus.

The manuscript is structured as follows: 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) and one synthetic dataset (IV) for *A.thaliana*, the well known model plant, and one, (dataset V) for two *H.sapiens* individuals, Table 1. Data for datasets I-III,V were generated 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 Computational resources

To run the HPC analysis pipeline nodes of the computational clusters at UPPMAX, equipped with dual 4-core (HPC1) and 8-core processors (HPC2) were used. Data and reference genomes were read and written to a parallel shared storage system. The main Hadoop test platform (Hadoop-in-the-Cloud) was deployed on a private cloud at UPPMAX using the OpenNebula [21] as a cloud manager. Each node in

this deployment was equipped with dual 4-core CPUs. The cluster was set up with Cloudera Hadoop distribution version 2.0.0-cdh4.4.0 [22]. For details on computational resources, see Supplementary material.

2.3 Analysis pipelines

We constructed two pipelines for identifying SNPs from short read data, one based on Hadoop and one on 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 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 (i.e. GATK [18]), we decided to create the analysis pipeline as simple as possible to be able to pass 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 in 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. For the exact workflow used one can refer to the code repository created for this work [**?**].

The equivalent Hadoop-based pipeline was implemented with the Crossbow. The input data and the indexed genome reference were copied to Hadoops 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 the 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 the Crossbow's preprocessor with our own, written in the MR-style, though sacrificing some generality[4]. To illustrate the benefits of this modification: For our 30-Gbase test dataset (approximately 45 GB of uncompressed FASTQ data) the preprocessing stage time shrunk from 3.3 hours to just under 2 minutes when run on 112 cores Hadoop cluster. The scripts for preprocessing stage are publicly available [**?**].

[4] we assume that the FASTQ data are BZIP archived, and delivered to the storage accessible by the Hadoop.

3 Results

3.1 Accuracy of pipelines

Since our HPC and Hadoop approaches use different SNP callers (Samtools and SOAP-snp correspondingly) we should not expect them to deliver perfectly matching SNPs lists, still we expect to capture and correctly identify the mutation. We tested the correctness just for the smallest dataset I. The mutation $C \to T$ on chromosome 4 at position 16702262 was successfully localized[23].

3.2 Scalability of HPC and Hadoop approaches

To demonstrate the scalability of the HPC approach we collected running time for dataset I, as a function of a number of cores used, see Table 2. As one can see the aligning process with the Bowtie scales linearly, but the SNP calling part is a definite bottleneck, revealing almost no scaling at all. 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 3.

Table 2: Calculation time for Dataset I executed on different number of cores on a node of HPC1 cluster.

N cores	timing, minutes		
	aligning	SNP calling[5]	total
1	38	37	75
2	19	36	55
4	10	36	46
8	5	36	41

Table 3: 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".

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
HPC1, 8 cores	41	96	157	307	596	1490	–

One of the attractive sides of using Hadoop is its almost linear scalability, i.e. calculation time linearly depends on the size of the dataset[19, 24], *regardless* the

[5] We do not consider here special tricks how to parallelize the Samtools analysis by chromosome, as could be done, as exemplified here http://www.biostars.org/p/48781/

Figure 1

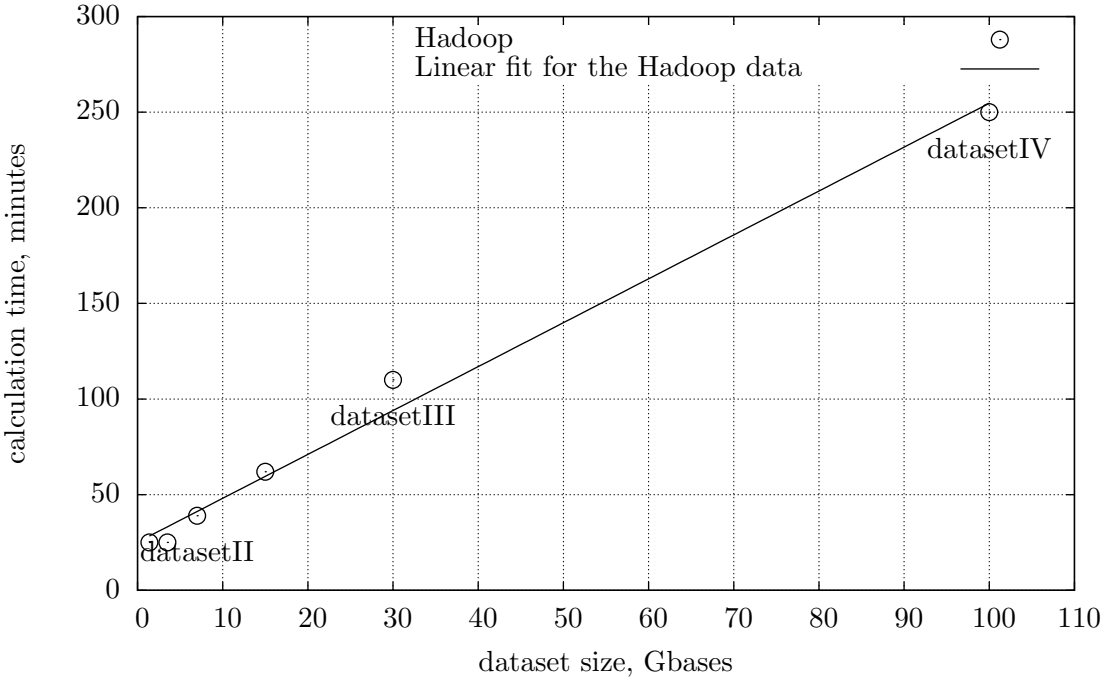


Figure 1: Calculation time depending on the dataset size for fixed size ($p = 56$ cores) of Hadoop cluster on the private Cloud. The linear fit based on least squares method are provided, revealing almost linear scaling.

scaling nature of the underlying program (Samtools, Bowtie etc.). Figure 1 shows the calculation time as a function of the dataset size for $p = 56$ cores Hadoop cluster. Dataset V was excluded since it is for *H.sapiens*, which has more than 20 times larger genome than *A.thaliana*. To stress the linear nature of scaling, both sets of points were fit into linear polynomial using least squares method.

3.3 Comparing Hadoop and HPC efficiency for different dataset sizes

Hadoop initially was designed to digest huge datasets[14, 15]. One can propose then, that the larger the dataset is, more suitable Hadoop becomes compare to HPC. In order to compare the "suitability" for different types of calculation platforms (Hadoop and HPC) each being run on different amount of cores, we constructed the following function:

$$F = T_p \times p,$$

Figure 2

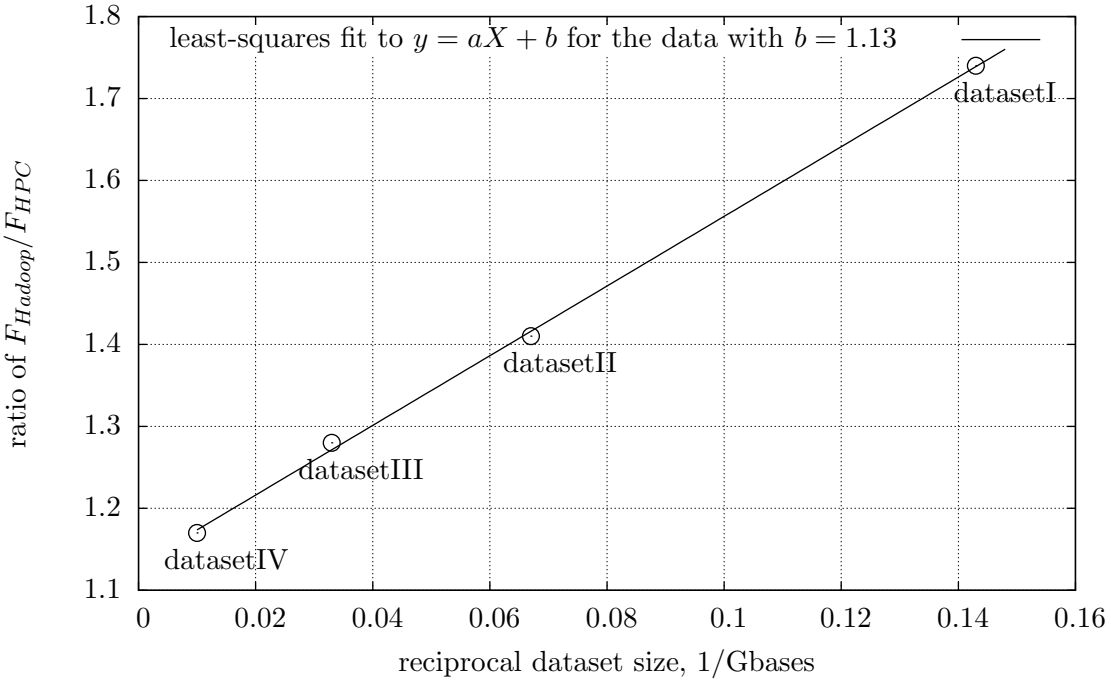


Figure 2: The ratio of the F_{Hadoop}/F_{HPC} as a function of a reciprocal dataset size in Gigabases. Calculations were carried out for $p = 56$ and $p = 8$ cores for Hadoop and HPC1 correspondingly. The points are fit in a line using least-squares method, what makes possible to make some careful predictions for the case of the *infinite* dataset size.

where T_p is a calculation time on p cores. Using the data from Table 3, we plot the ratio F_{Hadoop}/F_{HPC1}, keeping in mind, the closer the ratio to the unity, the closer Hadoops efficiency approaches to that of HPC.

The curve in Figure 2 is plotted for $p = 56$ cores Hadoop-in-the-Cloud cluster, and a HPC1 node ($p = 8$), and displays the ratio F_{Hadoop}/F_{HPC1} as a function of the *reciprocal* dataset size. The extrapolation to the zero of the X-axis tells 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 data becomes larger. With the linear extrapolation the ratio reads 1.13 ± 0.01, meaning that Hadoop running even in the *virtualized* environment is competitive with the HPC approach, which is being run in "bare metal", for the datasets greater than 100 Gbases (Dataset IV), which is a usual size for human genome sequencing, giving the average sequencing depth of about $30x$.

3.4 Comparing the network communication efficiency for Hadoop and HPC approaches

Network communication model for Hadoop has a major difference from the usual HPC cluster network architecture (n-tier tree) with the NAS or NFS attached storages. Namely, the effective bandwidth of the Hadoop network increases with the cluster size to increase [17], opposite to that of the HPC network, where the cluster grows results in network saturation and performance depletion. We provide a comparison how HPC and Hadoop network communication costs depend on the number of nodes involved, for a fixed dataset size, dataset IV.

Due to the fact of trivial parallelisation of the alignment process – the read-pairs are independent of each other, and can be aligned independently – one could try to involve more computation resources, i.e. split the initial data into chunks to proceed 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 travel through the network, potentially causing traffic jams, therefore increasing the communication costs, T_{comm}.

There are several program packages for short reads alignment designed with the MPI support [11, 13]. Authors report almost linear scaling up to 32 nodes for pair-ended reads[6]. However, i.e. Pmap package had been proved to function poorly on UPPMAX cluster for datasets larger than 20 Gbases, raising memory exceptions. We wrote custom state-of-art bash script involving existing Unix utilities [**?**] to use the HPC2 cluster network as efficiently as possible, 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 plot their ratio $T_{mapping}/T_{comm}$ as a function of a reciprocal number of nodes $1/N$. Such kind of measure is applicable to both HPC and Hadoop, however, the T_{comm} carries a different sense in both cases. For Hadoop the short reads in FASTQ format have to be preprocessed (involving nodes communication T_{comm}) to able to be run in the 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 the Crossbow to make it suitable for MR-style parallelisation. For the HPC approach, the T_{comm} involves the chunks travel from the sequencer delivery location to the local node scratches, where the actual mapping happens, and the travel of the aligned SAM 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. In its turn, the HPC approach is presented in two versions, which are based on a bit different strategies of the resource allocation.

Lets start from description of the Hadoop results, filled circles. 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 the Bowtie, used for mapping, provides a linear scaling between the mapping time and the dataset chunk size D: $T_{mapping} \propto D \propto 1/N$, see [6], and Figure 1. Since the ratio $T_{mapping}/T_{comm} \approx const$, one can

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

conclude that the $T_{comm} \propto 1/N$, meaning that the more nodes are involved, the faster the communication in the preprocessing stage is.

Lets now consider the curves for HPC2. The strategy named HPC SLURM[7] is as follows: the data from the delivery location is being split into chunks in *parallel*, which are *simultaneously* being pushed to the local scratches of the nodes, allocated by SLURM (open circles). One can see two distinct linear stretches, each with different tangent. One stretch is for the range from 4 to about 12 nodes the, and the another is from 12 up to 60. The former, horizontal stretch, is explained as for Hadoop – the more nodes is being 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[8] and unable to pass more data in a time unit, while the mapping time is still proportional to the chunk size: $T_{comm} \approx const, T_{mapping} \propto D \propto 1/N \rightarrow T_{mapping}/T_{comm} \propto 1/N$, i.e. a linear dependency, which one can observe on the plot. The transition area between two modes – saturated and unsaturated – has the next origin: each HDD on the local node can write the data at a speed about 100MB/sec $\approx$ 1Gbit/sec, i.e. 10 nodes will consume the data with the rate of 10Gbit/sec, what 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, what is the default SLURM behaviour, therefore the caption reads "HPC SLURM".

The scalability can be improved by overriding the default behaviour of the SLURM and 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 been used (58 nodes) the deviation from a linear speed-up 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 driven by the saturating the uplink cable with the throughput of 50Gbit/sec. The proposed strategies aimed to gain the maximum from the existing resources show, that even properly adjusted and tuned, HPC approach sooner of later starts to suffer from the network saturation.

At the same time, the HDFS keeps 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) free nodes to continue to investigate the scaling as in plot at Figure 3, but we do not expect any significant deviations, since the exposed behaviour is a generic for Hadoop with HDFS.

3.5 Usability aspects

At the present moment a popular way to steer the bioinformatic pipelines on the HPC resources is to use Galaxy[25], which provides a Web-based graphical user interface (GUI) to the bioinformatic programs, simplifying an experience for the end user. One of the alternatives for Hadoop could be the publically available Cloudgene[26],

[7] Simple Linux Utility for Resource Management

[8] 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

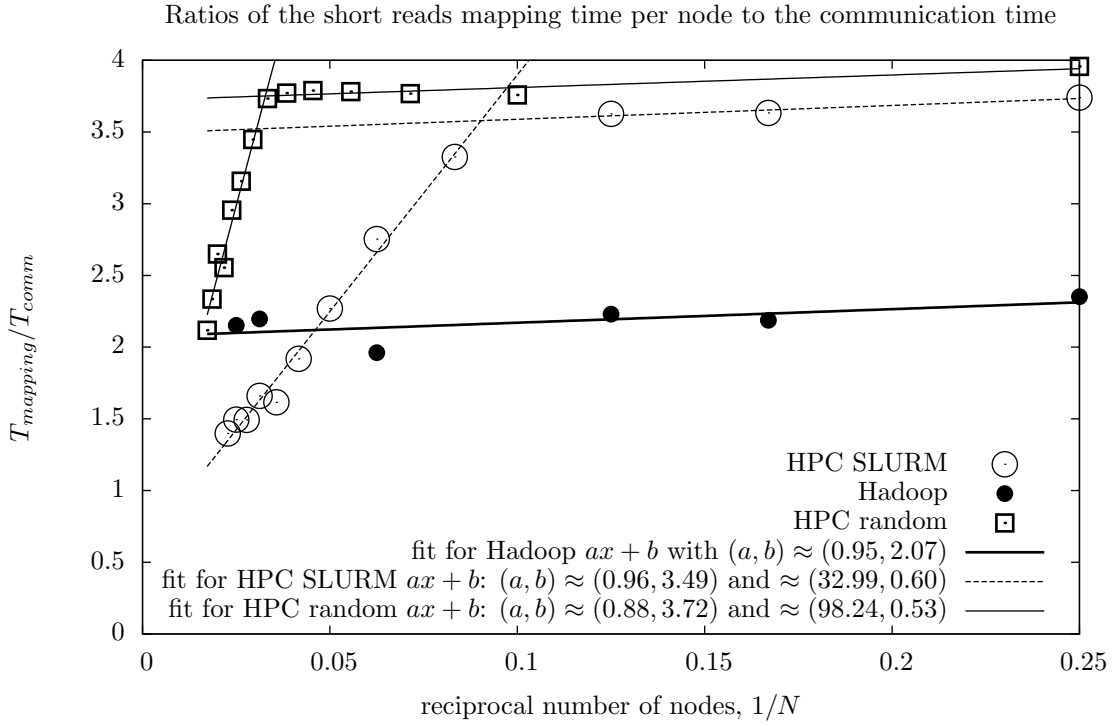


Figure 3: Ratios of the mapping time $T_{mapping}$ to the communication costs T_{comm} for HPC2 and Hadoop clusters for Dataset IV as a function of reciprocal cluster size $1/N$. Two HPC scenarios are shown as "HPC SLURM" and "HPC random", which correspond to standard SLURM behaviour and a modified one, where nodes are being allocated from random racks. Linear fit done with the least-squares method.

Table 4: Timings for mapping, and the ratio $T_{mapping}/T_{comm}$ for HPC2 and Hadoop clusters for Dataset IV. For "HPC random" approach data chunks have to be copied to the local scratches first, and the alignments (SAM files) copied back, while Hadoop keeps all the data inside HDFS. Hadoop needs to preprocess reads before the actual alignment stage, in order to be able to operate in a MR manner, also resulting in "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 nodes (cores)	Mapping time, minutes	$\frac{T_{mapping}}{T_{comm}}$	Number 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

which seems to be a very light-weight, and flexible Web-based solution for serve GUI for both public and private cloud, which we involved in our work. For the particular task in DNA-seq experiment, Cloudgene provides the intuitive interface making one easy to follow, Figure 4. The most of data managing job is done automatically, and the results can be downloaded to the client machine. Modular structure allows one easy to modify the source code to integrate to the existing computing centers architecture, Figure 4(a). For example the 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 stay locally, reducing amount of unnecessary risks.

4 Conclusions

In this communication we brought together different approaches to analyze the DNA-seq data in the bioinformatic study, in order to answer the question: which conditions a task should meet in order to make Hadoop a proper tool to use in favour of using HPC resources.

To increase the performance of the existing Hadoop software, the modification to the preprocessing stage of the Crossbow software were suggested. A state-of-art bash script was created to engage multiple nodes (up to 928 cores) in the HPC approach to align the Illumina-produced short reads with almost linear speed-ups.

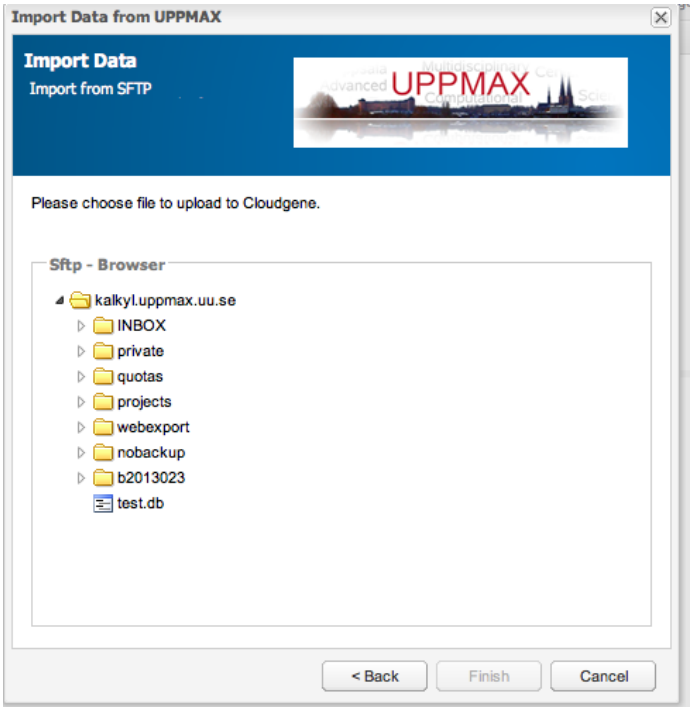
We concentrated ourselves on the Hadoop in the private cloud installation, Hadoop-in-the-Cloud, orchestrated by the OpenNebula. Picking the appropriate program settings we found out that DNA-seq dataset size larger than 100 Gbases is suitable for Hadoop, with competitive execution time compare to that of HPC. The scaling graphs (Figure 1) confirms the known fact for almost linear scalability of Hadoop. The extrapolation to the infinite dataset size (Figure 2) revealed, that HPC, however, provides the results faster, given the same amount of resources, than the Hadoop-in-the-Cloud.

Exploiting the embarrassingly parallel nature of the short reads mapping we used custom state-of-art bash script to engage up to 58 nodes (928 cores) of the HPC2 cluster to see the scaling relations between the ratio of the mapping time to the communication time as a function of a reciprocal number of nodes, Figure 3. Our results show that Hadoop with HDFS scales better than the network attached parallel storage commonly used in the HPC centers. In addition we show an example, how one can improve the performance of the HPC approach redefining the SLURM's default behaviour.

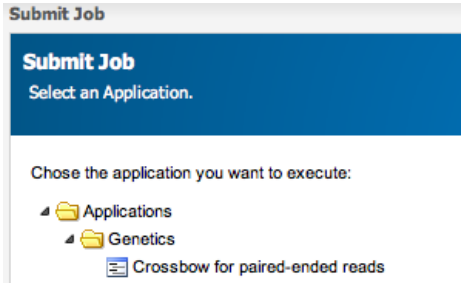
Finally, existing and easily transformed publically available web-based GUI, like Cloudgene (Figure 4), gives an opportunity to perform bioinformatic analysis on Hadoop for those who are not experts in Linux world. Modular structure of Cloudgene gives cluster staff an easy way to adapt it for the specific needs.

5 Acknowledge

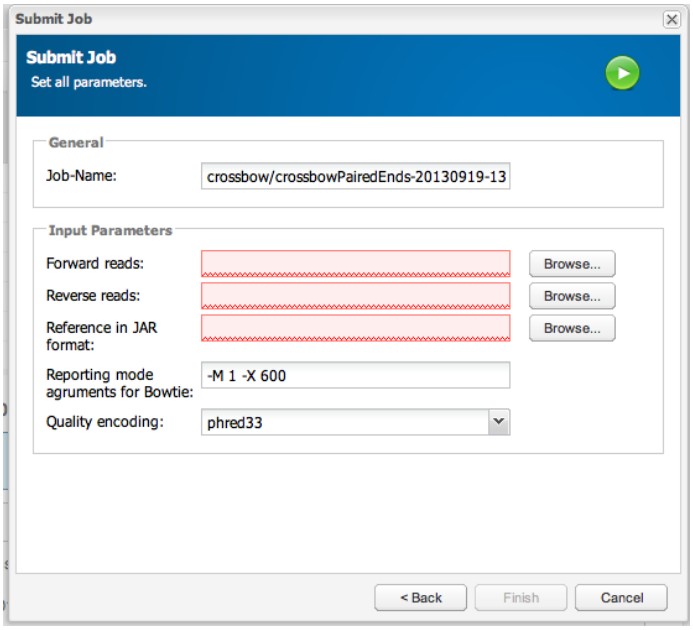
The computations were performed on resources provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under Project



(a) UPPMAX-adapted Cloudgene: browsing the users home folder



(b) Cloudgene: selecting a job to run



(c) Cloudgene: specifying job parameters

Figure 4: An example of a job setup with the Cloudgene, a web-based GUI wrapper, providing a smooth user experience even for novice users.

p2013023. We also greatly appreciate Pontus Freyhult and Peter Ankerstål at UPPMAX for explaining the effective storage usage. Great job was done by Jonas Hagberg at BILS, Stockholm, Sweden, by the adaptation of the Cloudgene to the local UPPMAX needs.

6 Supplementary material

6.1 Used 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. HPC1: The HPC analysis pipeline was run on a node from the Kalkyl [27] cluster, equipped with two quad-core processors Intel Xeon 5520 (clock frequency of 2.26 GHz; 1 MB L2 cache, 8 MB L3 cache), 24 GB of RAM and an Infiniband node-to-node network connection, and 10Gbit/s uplink. The data and reference genomes were read and written to a parallel shared storage system.
2. HPC2: Multinode short read mapping was performed on the Milou cluster[28] , equipped with dual 8cores CPUs 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.
3. Storage: Gulo[29] 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.

4. Our main Hadoop test platform was deployed on the private cloud at UPPMAX, using the OpenNebula [21] cloud management system. Each node in this deployment was equipped with dual quad-core CPUs (Intel Xeon 5420; clock frequency of 2.50GHz 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 [22].

References

[1] Michael 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] Samuel Lampa, Martin Dahlo, Pall Olason, Jonas Hagberg, and Ola 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] Nathan L. Clement, Quinn Snell, Mark J. Clement, Peter C. Hollenhorst, Jahnvi Purwar, Barbara J. Graves, Bradley R. Cairns, and W. Evan 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, MC Schatz, J Lin, M Pop, and SL 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] Open Nebula. http://opennebula.org.

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

[23] Korbinian Schneeberger, Stephan Ossowski, Christa Lanz, Trine Juul, Annabeth Hogh Petersen, Kare Lehmann Nielsen, Jan-Elo Jorgensen, Detlef Weigel, and Stig Uggerho Andersen. Shoremap: simultaneous mapping and mutation identification by deep sequencing. *Nat Meth*, 6(8):550–551, 08 2009.

[24] Luca Pireddu, Simone Leo, and Gianluigi Zanetti. Seal: a distributed short read mapping and duplicate removal tool. *Bioinformatics*, 2011.

[25] Belinda Giardine, Cathy Riemer, Ross C. Hardison, Richard Burhans, Laura Elnitski, Prachi Shah, Yi Zhang, Daniel Blankenberg, Istvan Albert, James Taylor, Webb Miller, W. James Kent, and Anton Nekrutenko. Galaxy: A platform for interactive large-scale genome analysis. *Genome Research*, 15(10):1451–1455, 2005.

[26] 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.

[27] Kalkyl cluster. http://www.uppmax.uu.se/the-kalkyl-cluster.

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

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