Bioinformatics, 34(6), 2018, 928–935
doi: 10.1093/bioinformatics/btx702
Advance Access Publication Date: 2 November 2017
Original Paper

OXFORD

Genome analysis

ViraPipe: scalable parallel pipeline for viral metagenome analysis from next generation sequencing reads

Altti Ilari Maarala[1,2,*], Zurab Bzhalava[3], Joakim Dillner[3], Keijo Heljanko[1,2] and Davit Bzhalava[3]

[1]Department of Computer Science, Aalto University, Espoo, Finland, [2]Helsinki Institute for Information Technology HIIT, Espoo, Finland and [3]Department of Laboratory Medicine, Karolinska Institutet, Stockholm, Sweden

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on June 9, 2017; revised on October 8, 2017; editorial decision on October 25, 2017; accepted on November 1, 2017

Abstract

Motivation: Next Generation Sequencing (NGS) technology enables identification of microbial genomes from massive amount of human microbiomes more rapidly and cheaper than ever before. However, the traditional sequential genome analysis algorithms, tools, and platforms are inefficient for performing large-scale metagenomic studies on ever-growing sample data volumes. Currently, there is an urgent need for scalable analysis pipelines that enable harnessing all the power of parallel computation in computing clusters and in cloud computing environments. We propose ViraPipe, a scalable metagenome analysis pipeline that is able to analyze thousands of human microbiomes in parallel in tolerable time. The pipeline is tuned for analyzing viral metagenomes and the software is applicable for other metagenomic analyses as well. ViraPipe integrates parallel BWA-MEM read aligner, MegaHit De novo assembler, and BLAST and HMMER3 sequence search tools. We show the scalability of ViraPipe by running experiments on mining virus related genomes from NGS datasets in a distributed Spark computing cluster.

Results: ViraPipe analyses 768 human samples in 210 minutes on a Spark computing cluster comprising 23 nodes and 1288 cores in total. The speedup of ViraPipe executed on 23 nodes was 11x compared to the sequential analysis pipeline executed on a single node. The whole process includes parallel decompression, read interleaving, BWA-MEM read alignment, filtering and normalizing of non-human reads, De novo contigs assembling, and searching of sequences with BLAST and HMMER3 tools.

Contact: ilari.maarala@aalto.fi

Availability and implementation: https://github.com/NGSeq/ViraPipe

1 Introduction

The metagenomics (Thomas *et al.*, 2012) in the era of Next Generation Sequencing (NGS) enables the complete sequencing of all microbiological sequences that may be present in a sample. The human microbiome (i.e. microbiota), defines the collection of micro-organisms that reside in the human body (Rogers and Bruce, 2012). Our microbiome is recently found to reflect our health, and associate to infections and many diseases (Hall *et al.*, 2017; O'keefe and Greer, 2011; Robinson and Pfeiffer, 2014).

The viral fraction of human microbiome, i.e. virome (Wylie *et al.*, 2012a), constitute only a small part of human microbiota, but the proportion and composition of viruses seem to vary in diseased individuals (Thomas *et al.*, 2012; Wylie *et al.*, 2012b). Studies on viral metagenomics require unbiased sequencing of all DNA in

© The Author 2017. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

biospecimens, in contrast to studies of bacterial communities where the conserved 16S rDNA is typically targeted. Viruses usually have smaller genomes than other microbes and virus-related sequences will usually constitute only a small fraction of all sequences.

NGS technolofy brings new opportunities for metagenome analyses and enables analyzing of microbial DNA as a part of microbial community directly from its natural environment, rather than from laboratory cultured microbes. Viral metagenomics is nowadays routinely used for virus detection and is commonly used for discovering of new viruses (Arroyo Mühr *et al.*, 2017, 2015a,b; Bzhalava *et al.*, 2012, 2013, 2014; Smelov *et al.*, 2016). NGS provides an opportunity to perform a large-scale analysis of all infections that are present in human biospecimens. Thus, metagenomics has now the potential to further our knowledge of the role of viruses in human diseases. However, this raises a great computational challenge for performing genome-wide analyses with ever-growing data volumes in tolerable time.

The concurrent computation is the growing trend and the concurrency is increasing at multiple levels: the number of computing cores in the processor, processors on the computer, and computers in the cluster. The amount of NGS data worldwide is predicted to double every 7 months (Stephens *et al.*, 2015). However, the current genome analysis pipelines are typically developed utilizing mixture of command line tools together with the existing sequential algorithms making analyses computationally inefficient, inflexible and not executable on distributed systems and cloud computing environments. Hence, the analytical tools and algorithms need to conform to exploit all this parallelism efficiently. Moreover, the existing sequential algorithms and pipelines are run on expensive centralized systems whilst cloud computing environments provide cost-efficient computing power on demand.

In our previous work, we have been focusing on parallel processing of human NGS data formats on Hadoop framework with Hadoop-BAM (Niemenmaa *et al.*, 2012) and SeqPig (Schumacher *et al.*, 2014). In this work, we design and implement a parallel Apache Spark based pipeline, ViraPipe, for analyzing viral metagenomes from hundreds of NGS samples. ViraPipe includes read aligner with standard BWA-MEM algorithm, *De Novo* genome assembler with MegaHit (Li *et al.*, 2016), NCBI BLAST and HMMER hmmsearch (Eddy and Pearson, 2011) tools for discovering nucleotide and protein sequences, as well as, data filtration and normalization tools, and SparkSQL based interface for querying the results. Moreover, we run the metagenome analysis experiment on real NGS datasets generated from human samples and evaluate the results.

2. Materials and methods

2.1 Distributed and parallel computation

Storing and processing of genomic data requires large amount of high performance storage space, working memory, computing power and network capacity. Distributed computation frameworks, file systems and databases have been evolving while the price of storage and memory has been decreasing and now it is economically viable to move on to computing clusters and cloud computing. Cloud computing frameworks enable scalable, reliable, efficient and relatively low cost computing in a server clusters. Cloud services provide infrastructures for deploying computing cluster easily and flexible way. Parallel data analysis with multiple distributed computer nodes brings huge performance advantage compared to a standalone machines.

2.2 Parallel characteristics of genomic data

Back in the days when widely used algorithms and methods for computational genomics were designed, the parallel computation was not properly considered. Some of these are challenging to parallelize such as BWA, Bowtie, BLAST, Hidden Markov Models and *De Novo* assembly algorithms. However, data parallel processing is one potential choice for parallelizing genome analysis pipelines without rewriting the existing algorithms. Data locality in multiple nodes of the computing cluster speeds up the processing as data is processed in parallel in local memory without reloading or moving any data. The existing general genomics file formats are not designed for distributed file systems and especially binary formats, such as BAM, BCF and BED are not distributable without external tools. However, Hadoop-BAM (Niemenmaa *et al.*, 2012) library offers functionalities to handle distributed BAM and BCF files on Distributed Hadoop Files System (HDFS) in parallel and also in-memory with Spark. Genomic data is usually parallelizable per read sequence, chromosomes and regions by gene locus, which gives opportunity to distribute the input data in most processing phases. For example, NGS data can be distributed for read alignment if reference assembly methods are used such as BWA (Li and Durbin, 2009) or Bowtie.

Index databases become challenging when distribution has to be considered. For example, the index of the reference genome has to be provided as whole at the read alignment phase for aligning reads globally to a reference genome. Thus, the reference index is replicated to every node in the computing cluster. The index database for the local BLAST alignment can be distributed and queried in parallel if the input data is relatively small and the input query is replicated. Otherwise, the queries can be segmented and distributed to replicated database for parallel search (Wang and Mu, 2003).

2.3 Apache Hadoop and Spark

The popular distributed computing framework, Apache Hadoop (https://hadoop.apache.org/) and the MapReduce (Jeffrey and Sanjay, 2004) programming model are based on parallel data processing of distributed data. MapReduce is intended for processing large datasets. In particular, it is programming model for Big Data processing in Hadoop framework, which can perform a wide variety of real-world computing tasks. Computing with MapReduce is performed in parallel on distributed nodes typically deployed in Hadoop computing cluster. MapReduce has advantages as it provides easy to use template for programmers to perform parallel computing tasks. Furthermore, MapReduce is a powerful tool for the large scale fault tolerant data analysis.

Apache Spark (Zaharia *et al.*, 2010) is open-source framework developed for iterative parallel in-memory data processing on computing clusters whereas MapReduce on Hadoop is based on acyclic batch processing and thus deficient with iterative jobs and interactive analysis. Spark accelerates large-scale data processing with iterative analysis where same working sets of data are cached and reused several times within same job. Computation in Spark is based on Resilient Distributed Dataset (RDD) (Zaharia, 2012), which is distributed and cached to working memory of multiple worker nodes in a cluster. RDD is processed in partitions within parallel tasks managed by YARN node managers. Each node then assigns executor for local tasks which are run in parallel on multiple cores inside a node. Spark supports reading and writing data to HDFS and Apache HBase, which are provided with most Hadoop distributions. Spark can be programmed with Scala, Java, Python and R programming languages, and it includes extensions for parallel graph

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

processing, machine learning and data streaming. Figure 1 shows typical data reduction process in genomic data context in Hadoop framework.

2.4 Hadoop and Spark based genomic data processing frameworks

Hadoop-BAM is a library originally developed for processing NGS data formats in parallel with both Hadoop and Spark. It includes Hadoop Input/Output interface for distributing genomics file formats into HDFS and tools for e.g. sorting, merging and filtering of genomic file formats. Currently, supported file formats are: BAM, SAM, CRAM, FASTQ, FASTA, QSEQ, BCF and VCF. Hadoop-BAM (Niemenmaa *et al.*, 2012) is widely used in parallel genome analytics frameworks and libraries such as GATK4 (https://github.com/broadinstitute/gatk), Adam (https://github.com/bigdatagenomics/adam), Halvade (Decap *et al.*, 2015), Seal (Pireddu *et al.*, 2011) and SeqPig (Schumacher *et al.*, 2014).

GATK (McKenna *et al.*, 2010) is a software package for genomic data analysis developed by the Broad Institute which has been originally focusing on large scale germline variant discovering of human genomes, but nowadays it provides also tools for somatic variant analysis workflows and also copy number analytic workflows are under development. The current GATK4 (https://software.broadinstitute.org/gatk/download/alpha) version is partly developed on Apache Spark for supporting more scalable and rapid exploratory genomic studies. The Broad Institute has also established open source FireCloud platform for managing, sharing and analyzing genomic data. GATK4 implements also parallel BWA aligner, but it lacks specific filtering features and the FASTQ input format for metagenomic analysis.

ADAM (http://bdgenomics.org/projects/adam/) is Apache Spark based genome analysis toolkit developed in UC Berkeley for exploring genomic data. ADAM includes basic tools for genomics file transformations, k-mer counting, and allele frequency computation on Apache Spark cluster.

Halvade (Decap *et al.*, 2015) is a scalable Hadoop MapReduce framework for parallel variant calling based on the Broad Institute's best practices variant discovery workflow. Halvade uses MapReduce jobs for parallel BWA read alignment and for discovering variants with GATK's variant calling pipeline. SealSeal (Pireddu *et al.*, 2011) is a software suite developed in CRS4 for processing sequencing data based on Hadoop framework and is written in Python. It includes basic tools for aligning reads to a reference genome, identifying duplicate reads, sorting the reads and read quality controlling.

CloudBrush (Chang *et al.*, 2012) and Contrail (https://sourceforge.net/projects/contrail-bio/) are parallel *De Novo* assemblers developed on Hadoop MapReduce framework. The both are based on de-bruijn string graph processing. Spaler (Abu-Doleh and Çatalyürek, 2015) is a *De Novo* de-bruijn graph assembler based on Spark framework and GraphX API. However, the Spaler software is not publicly available currently.

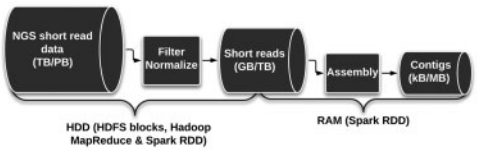

Fig. 1. Data reduction process

2.5 Viral metagenomics workflow

The workflow is preceded by read alignment process where billions of short read sequences from donor sample are aligned to a reference genome. The amount of NGS data coming from the sequencer machine depends on the size of the donor genome, used sequencing method and parameters given to a sequencer. For example the maximum read data output of Illumina NextSeq Series sequencing platform is 120 gigabases consisting of 150 base pair long DNA or RNA fragments (reads). These short reads are aligned to human reference genome for filtering out human mapped reads and for assembling metagenomes from the unmapped reads. The reference alignment process requires every read to be aligned against every position in the reference genome where the reference sequence is three billion base pairs long for the human reference (resulting to 3 GB of reference data). There are a few standard tools for relatively fast sequence alignment such as Burrows-Wheeler transformation based BWA, Bowtie and SOAP aligners.

NGS technology provides relatively cheap and rapid Whole Genome Sequencing (WGS), exome sequencing, and targeted sequencing of DNA and RNA where millions of short reads are sequenced from host sample in parallel. Metagenome analysis pipelines typically implement re-sequencing of viruses or bacterial DNA from donor NGS samples for further downstream analysis. A bioinformatics pipeline for analyzing viral metagenomic datasets starts typically with aligning raw sequencing reads against a human reference genome, e.g. using the BWA aligner (http://bio-bwa.sourceforge.net/bwa.shtml). The unmapped read sequences are then normalized using e.g. digital normalization (http://ged.msu.edy/papers/2012-diginorm) to discard redundant sequences. The normalized reads are assembled with *De Novo* assembly tools such as SOAPdenovo, SOAPdenovo-Trans (http://soap.genomics.org.cn/), Trinity (http://trinityrnaseq.github.io), IDBA-UD (http://i.cs.hku.hk/ãlse/hkubrg/projects/idba_ud/) and MegaHit (https://github.com/voutcn/megahit). The use of several assembly algorithms and re-assembling of all singleton reads to contigs can be used to improve assembly results. The assembled contigs are queried from human genomic database with BLAST and contigs having identity greater than 95% over 75% of their length to human, bacterial, phage and vector DNA are removed from further analysis. The assembled contigs are then subjected to taxonomic classification by comparing them against BLAST nt and nr databases to classify them as i) previously known sequences, ii) related to previously known sequences or iii) unrelated to any previously known sequences.

Zhou et al. propose MetaSpark (Zhou *et al.*, 2017) framework for fragment recruitment, i.e. finding the fraction of short read sequences that are presented in the reference genome. This process requires read alignment against already assembled metagenomes and MetaSpark implements FR-HIT algorithm on Spark for read alignment. FR-HIT algorithm is based on basic Smith-Waterman local alignment algorithm and FR-HIT is developed especially for fragment recruitment purposes.

EBI Metagenomics (Mitchell *et al.*, 2016) is online service for analysing and archiving metagenomic data. Their pipeline does not perform reference alignment and is used mainly for analyzing environmental samples rather than human samples and viruses. The pipeline itself is based on identification of protein coding sequences with hidden markov models and searching of matches from multiple databases.

ViraPipe is specifically tuned for mining virus related genomes, as well as detection of novel potentially virus-related sequences in metagenomic datasets generated by NGS technologies.

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

2.6 Implementation

ViraPipe utilizes data parallel computation strategy as genomic data can be processed in partitions at many levels (divided by chromosomes, chromosomal regions and short read partitions) and this approach enables us to use existing bioinformatics tools and algorithms in the pipeline. The sequential methods and tools have been developed for the most general processing phases such as read alignment, *De Novo* assembly and local alignment search with BLAST. ViraPipe is implemented on Apache Spark and Hadoop with HDFS file system as this provides flexible framework for scalable and efficient parallel data processing and distributed data management. Key requirements are partitioning of data for parallel execution with existing genomic data formats, and piping of multiple tools and algorithms together with minimal I/O operations. Moreover, low latency for reading data from HDFS to in-memory RDD and writing back to HDFS is important as bioinformatics pipelines usually process thousands of separate files as well as big files together. This sets high performance requirements for the deployment cluster disk I/O, memory allocation, data warehousing and networking.

ViraPipe aligns distributed FASTQ formatted read chunks with BWA-MEM from the HDFS read data partitions in parallel, and utilizes Hadoop-BAM library for parallel I/O operations and read filtering. Parallel read normalization is performed to remove redundant sequences including similar k-mers for viral genome *De Novo* assembly. ViraPipe integrates MegaHit assembler (https://github.com/voutcn/megahit) for assembling normalized non-human reads to contigs in parallel per sample. The well-known BLAST search is executed in parallel per contig partitions to query the assembled contigs from the BLAST human genomic and nt databases. The BLAST results are filtered and classified by taxonomies. Finally, the HMMER search is executed to find contigs matching to viral protein sequences in vFam database. Figure 2 describes the workflow of the pipeline at high level.

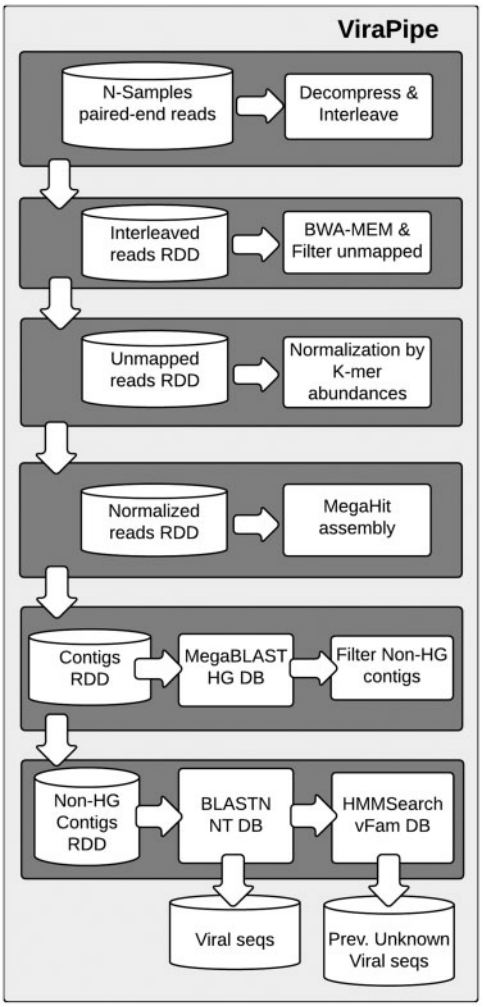

Fig. 2. ViraPipe workflow

2.6.1 Pre-processing NGS input data

The sequenced paired-end reads are first decompressed and interleaved by in parallel by sample and the interleaved reads are distributed to HDFS in FASTQ format using the Hadoop-BAM I/O library. The indexed reference genome is replicated to each computing node for the BWA alignment process.

2.6.2 Reference alignment and filtering

Parallel BWA-MEM based aligner is implemented with JBwa (https://github.com/lindenb/jbwa) library which binds the BWA-MEM library through Java Native Interface to Spark implementation. Distributed FASTQ read data is first loaded from the HDFS to Spark RDD. A Spark task loads the reference genome index and read data partitions are aligned to reference with JBwa library in parallel tasks. During the same task, the unmapped reads are filtered and the SAM formatted sequences are transformed back to FASTQ format for further processing. In addition, BWA output can be stored in SAM/BAM (Li *et al.*, 2009) format into HDFS.

2.6.3 K-mer normalization

Read normalization removes highly redundant sequences by searching and discarding reads containing similar k-mer sequences. Basically, the process performs digital normalization (Brown *et al.*, 2012) and filters out reads including k-mers that have abundance above given cutoff threshold. The normalization is executed on Spark over parallel read data partitions. At the end of this phase, the normalized reads are grouped by sample for use in *De Novo* assembly phase.

2.6.4 De novo contigs assembly

Normalized FASTQ reads are assembled in parallel per sample. MegaHit is executed in parallel Spark tasks. To increase the concurrency MegaHit uses multiple threads in a task to assemble contigs from each sample. After every task execution the results are stored into HDFS. The same parameters are supported as native MegaHit implementation provides.

2.6.5 BLAST search for viral genomes

Assembled contigs are searched with BLAST from the database which is replicated in the local file system of each cluster node. The input contig data is read from the HDFS in FASTA format produced by MegaHit assembler and the contigs are repartitioned with Spark RDD for parallel BLAST search. The same options and output formats are supported as native BLAST implementation provides. With fewer partitions, more threads are used to harness all of the cores as BLAST is queried per file partition.

First, all the contigs are searched from the BLAST human genome database with megaBLAST and the non-human contigs are filtered and kept. The human matching contigs with identities below given threshold are also kept. The candidate contigs are searched

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

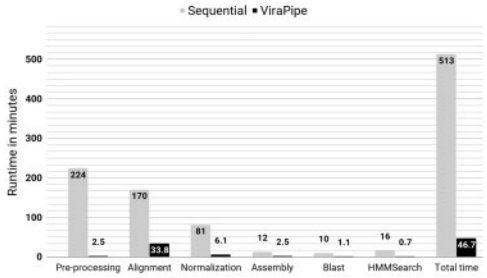

Fig. 3. ViraPipe performance compared to the sequential pipeline with 13 samples (105.5 GB) dataset

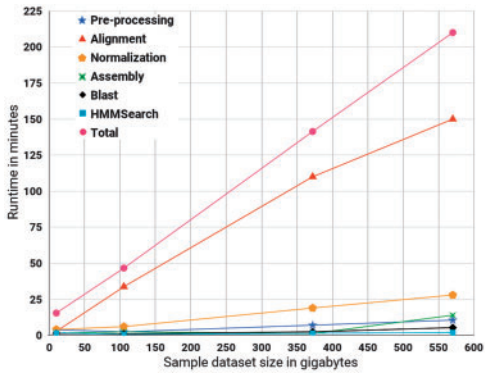

Fig. 4. Scalability of the pipeline in respect to the sample dataset size

from BLAST nt database and viral sequences are classified by taxonomy from the results. All other non-matching contigs are classified to 'unrelated to any previously known sequences'.

2.6.6 HMMER3 search for viral genomes

HMMER3 (Mistry *et al.*, 2013) (hmmsearch algorithm) is used to search homologous protein sequences from the assembled contigs which are classified as 'unrelated to any previously known sequences' by NCBI BLAST search. The contigs are queried with hmmsearch from vFam reference database (http://derisilab.ucsf.edu/software/vFam) that includes viral profile HMMs ('vFams') annotated as viral proteins collected from RefSeq database (Skewes-Cox *et al.*, 2014). Similar to BLAST search, the partitioned FASTA files are loaded to Spark RDD and hmmsearch is executed in parallel per contig data partition. Multi-threading is supported by hmmsearch to increase concurrency.

3 Results

The scalability is the focus of our experiments. First, we run an experiment on a sequential pipeline and ViraPipe and compare the results in Figure 3. Second, we run an experiment with ViraPipe by varying the amount of samples (Fig. 4). Third, we run ViraPipe by varying the amount of parallel Spark executors (Fig. 5) on 13 samples (105.5 GB) dataset. Finally, we evaluate all the processing phases described in Table 2.

3.1 Patients and samples

The metagenomic sequencing datasets were generated by Next Generation Sequencing (NGS) technologies applied to human biospecimens originating from several different patients groups and were published elsewhere (Arroyo Mühr *et al.*, 2015a,b; Bzhalava *et al.*, 2012, 2013, 2014; Smelov *et al.*, 2016). In brief, all of these analyses were designed to investigate the presence of viral sequences or other microorganisms in human samples from individuals who developed diseases or from matched healthy control subjects.

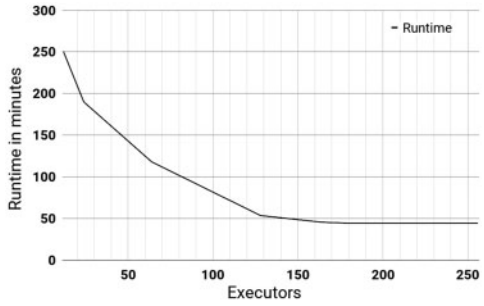

Fig. 5. Scalability of the pipeline in respect to the number of executors with 13 samples (105.5 GB) dataset

For testing the scalability, datasets of two, 13, 96 and 768 human samples are included totaling uncompressed dataset sizes of 10.1, 105.5, 372.1 and 570 GB respectively. The sample data has sequenced from Swedish individuals during the projects between the years 2011–2015. All the samples has sequenced from different individuals. The NGS read data of every sample was aligned to GRCh19 human reference genome in the experiments.

3.2 Cluster setup

The experiments were run on dedicated Hadoop cluster maintained by the department of Laboratory Medicine at Karolinska Institutet. The cluster comprises 24 nodes including 2× Intel Xeon E5-2690v4 @2.6 GHz CPUs (56 cores per node) and 256 GB of RAM in each. 23 nodes were deployed as Spark worker nodes for computation, totaling 1288 CPU cores and 5.12 TB of RAM. One node was deployed as Hadoop Namenode and for cluster management purposes. Each computing node has 2× 960GB SSD disks for local file-system and operating system. HDFS uses JBOD disk array totaling 1.5 petabytes of storage space where 15× 6TB HDD disks are connected to each node. The nodes were connected with 10 GB full duplex Ethernet adapters. Linux Ubuntu 16.04 was used as an operating system on each computing node. The Spark configuration was optimized for each processing phase (Spark job) in the pipeline separately. The HDFS data partition size and memory usage of the parallel tasks set the requirements for the number of executors, executor cores and the executor memory. The HDFS partition size is constant 128 MB and the memory allocation for BWA-MEM alignment task was 30 GB at maximum, thus, the parallel BWA execution is limited to 170 tasks (each task was run on single core) with 5.12 TB of cluster RAM. 170 executors with 30 GB of memory were found optimal for the alignment phase regarding the experiment (Fig. 5) where runtime on 105 GB dataset (13 samples) with increasing the number of executors was benchmarked. Increasing the number of executor cores does not improve the performance as available memory restricts the number of parallel tasks. The same strategy was used with other jobs in the benchmarked pipeline and the configuration is shown in Table 1. The dynamic Spark executor memory allocation was found suboptimal in preliminary tests. The sequential pipeline was set up for comparing the performance with the ViraPipe. The sequential pipeline uses the same bioinformatics tools and databases as ViraPipe except the read normalization is done with Khmer (https://github.com/dib-lab/khmer) tool. The Khmer is not utilizing multithreading as does the other tools in the pipeline. The pipeline

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

was run on a single node with 256 GB of memory on 56 cores in total using HDD storage. The sequential pipeline is available at https://github.com/NGSeq/ViraPipe/tree/master/scripts.

3.3 Evaluation

Table 2 shows the ViraPipe runtimes and input data sizes in each processing phase with different datasets as well as the total runtime, pipeline throughput, number of samples, number of total reads, read lengths and size of discovered contig data.

As a baseline for the evaluation, we run the experiment on 105.5 GB (13 samples) dataset with sequential pipeline and compare the results with ViraPipe in Figure 3. The total running time for the sequential pipeline is 513 min while ViraPipe performs in 46.7 min.

In the second experiment, we evaluate the runtime of the ViraPipe in function of sample dataset size. The results (Fig. 4) show that ViraPipe runtime increases linearly in proportion to dataset size.

Decompression and interleaving of reads took from 3.9 to 10.6 min depending on the number of samples and compressed input data size.

The BWA-MEM read alignment took 2.8 min with the smallest dataset including two samples (10.2 GB from 39.3 million reads) and 150 min with the largest dataset including 768 samples (570.9 GB from 2.07 billion reads). The read alignment process consumed nearly 70% of the total execution time when dataset size was hundreds of gigabytes. With smaller datasets, read alignment time was not that dominant and BLAST search became more expensive. This depends on the sample data, the more contigs are found the more time is spent on the BLAST search. For example, from dataset of two samples (10.1 GB) 150 kB of contigs were we assembled whilst 58 kB was assembled from 96 (105.1 GB) samples (Table 2). With two and 13 samples the BLAST search took 1.4 and 1.1 min while alignment took 2.8 and 33.8 min, respectively.

The normalization runtime grows linearly as the function of dataset size showing good scalability. The normalization reduces the size of data to 10–25% of the original unmapped reads.

Assembly time varies between 1.3 and 14 min (Table 2). The runtime depends on the dataset size of a sample as assembly is performed in parallel per sample. Thus with greater dataset size assembly can take less time than with smaller datasets.

The BLAST search time varies between 1.1 and 5.4 min depending on the size of assembled contig data. 1.4 min running time was achieved with 58.4 kB of contig data whereas 5.4 min was achieved with 2.3 MB of contig data. Human genomic database was queried with the megaBLAST. Input data was partitioned into 100 partitions and each partition was queried in parallel with multi-threaded BLAST algorithm.

HMMER3 search time varies between 0.7 and 2 min with same contig datasets as BLAST search was performed with. The data was partitioned into 100 partitions and each partition was queried in parallel executors with hmmsearch algorithm using 10 threads.

Finally, we run an experiment by increasing the amount of Spark executors with 13 samples dataset (105.5 GB). The total runtime in Figure 5 shows that it decreases linearly in proportion to the number of executors until 170 executors are reached.

Table 1. ViraPipe experiment configuration

Job	Executors	Executor memory	Tool parameters
Pre-processing	80–500[a]	10GB	–
Alignment (BWA)	170	30GB	mem -p
Normalization	80–250[a]	20GB	−k16 -C15
Assembly (MegaHit)	23–250	20–220GB[a]	−t10-56[a] –12
Blast	100	50GB	–threshold 70 –num_threads 12 –word_size 11 –gapopen 0 –gapextend 2 –penalty -1 –reward 1 –max_target_seqs 10 –evalue 0.001
HMMSearch	100	50GB	–noali –cpu12

[a]The amount is increased inversely proportional to the number of analyzed samples.

3.4 ViraPipe assessment

For the metagenomics analysis assessment we run both a sequential pipeline and ViraPipe with a single sample dataset. The ViraPipe relies on data parallel computation where the NGS reads are partitioned into blocks and processed in parallel. Thus, the BWA alignment is performed in parallel per sample read data partition, where each sample is processed separately and the reads of the sample are aligned in parallel. This reduces the parallelism to the number of partitions generated per sample but is mandatory for avoiding false mappings which occurs if the sample reads are merged before the alignment. The normalization is done also in parallel per read

Table 2. Runtime measurements are reported in minutes at different processing phases

Samples	2	10.1 GB	13	105.5 GB	96	372.1 GB	768	570.9 GB
	Runtime (min)	Input size	Runtime (min)	Input size	Runtime (min)	Input size	Runtime (min)	Input size
Pre-processing	3.9	1.5 GB	2.5	22.3 GB	7.1	135.1 GB	10.6	189.3 GB
Alignment	2.8	10.1 GB	33.8	105.5 GB	110	372.1 GB	150	570.9 GB
Normalization	4.1	3.8 GB	6.1	5.1 GB	19	31.4 GB	28	137.1 GB
Assembly	1.5	0.326 GB	2.5	0.389 GB	1.3	6.7 GB	14	10.2 GB
Blast	1.4	150 kB	1.1	58.4 kB	2.5	278 kB	5.4	2.3 MB
HMMSearch	1.8	150 kB	0.7	58.4 kB	1.5	278 kB	2	2.3 MB
Total runtime	15.5		46.7		141.4		210	
Throughput (reads/s)	42.3k		139.3k		193.6k		164.2k	
Total input reads		39.3 M		390.4 M		1642.2 M		2068.6 M
Read length		16–150 bp		35–150 bp		101 bp		101 bp

Note: Input data size in each phase is reported in bytes. Total input reads describes the total amount of sequenced reads, the read length is the length of single read in base pairs.

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

partition and this increases the amount of assembled contigs compared to output of the sequential pipeline. That is, the parallel normalization does not reduce the amount of reads as much as sequential normalization, thus, the ViraPipe finds a greater number (12.5%) of similar sequences than the sequential pipeline. With the test dataset, ViraPipe found 12.5% more non-human sequences.

4 Discussion

The results in Figure 3 show that overall speedup is 11× with 13 samples (105.5 GB) dataset. It should be noted that the sequential pipeline pre-processing phase includes read filtering operations whilst ViraPipe does the read filtering within the alignment phase in parallel. The ViraPipe pre-processing phase includes decompression and read interleaving phases.

When the number of samples and dataset size increases the scalability gets close to linear. Suboptimal performance with small amount of samples (2 and 13 samples) indicates that data is divided in too few partitions to harness all the computing power of available cores. During the experiments, the load balancing was not equal amongst the executors, thus some of the parallel tasks run longer increasing the total running time, especially in BWA-MEM alignment phase the amount of mapping reads affects to task running time greatly. More cores can be harnessed by decreasing the HDFS block size (defaults to 128 MB) especially with small datasets. The result of too big partition size can be observed from runtime of pre-processing phase: the runtime was shorter with 96 samples than with 2 samples (Table 2). Each sample is decompressed from read files in parallel. If the read files are compressed into size of configured HDFS blocks, the decompression would be considerably faster also with fewer samples.

The read sequence length can affect the performance of BWA algorithm, thus, the experiments with variable length reads may introduce a slight runtime overhead as partitions can include different amount of reads making task load balancing unequal. Also, the JNI interface can slow down the alignment performance slightly. However, the overall runtime grows linearly and aligner is performing great.

The normalization performance depends on the amount of the abundant k-mers and the number of data partitions, which is shown in the results (Table 2): 10 GB dataset (two samples) produced 82 partitions while 105.5 GB (96 samples) dataset produced 850 partitions. However, the runtime increases only 1.5 times, indicating that concurrency was not optimal with the smallest dataset, and thus, data should be processed in smaller partitions.

MegaHit assembler is executed in parallel per sample and assembly phase scales linearly. The assembly time varies and depends greatly on the number of samples and the amount of assembled contigs. However, the runtime of the 96 samples was shorter than with 13 samples resulting to 278 and 58.4 kB of assembled contig data respectively even though input data size was almost 20 times larger with 96 samples. That is, the contigs are assembled in as many partitions as is the number of original samples. Thus, with the small number of samples it is recommended to increase the number of MegaHit threads to increase the concurrency. However, the assembly phase consumes only a fraction of the total computing time.

The BLAST search is computationally intensive phase especially against the BLAST nt database which is 50 GB in size. The query time depends greatly on the query size which can be seen from the Table 2. In the experiments, the contig data is partitioned into 100 partitions as default. The length of the assembled contigs can affect to performance as long individual contigs can not be splitted for searching. However, in our experiments the assembled non-human contig length stays relatively small (under 5000 bp) thus, the query length is not showing any significant bias to the results. It was noted during experiments that too few memory increases the Blast runtime dramatically. By increasing the executor memory from 20 to 60 GB, the running time decreased to ten fold.

HMMER3 search runtime shows linear scalability. The hmmsearch was performed in parallel per RDD partition against vFam database which is relatively small (500 MB) compared to Blast nt database (50 GB). To report, we run also HMMER3 search against vFam database with 3.9 GB dataset and hmmsearch processed it in 22 min in parallel.

Scalability in respect to the number of Spark executors (Fig. 5) shows linearity until the amount of 170 executors is reached. That is, the total working memory of the cluster limits the amount of parallel execution of tasks. For example, at least 30 GB of memory is required for alignment phase when HDFS partition size is 128 MB, e.g. 170 executors each allocated with 30 GB of memory results to 5.1 TB of memory, which is near the total memory of the cluster.

A recent study of Unified Parallel Programming based alignment tool reports that their solution aligns 246 million paired end reads in 16.64 min (Gonzalez-Domínguez *et al.*, 2016) with CUSHAW3 aligner, which is reported to be fastest but not the most sensitive read aligner in (Lin *et al.*, 2011). Puckelwartz et al. (Puckelwartz *et al.*, 2014) performed variant calling on 240 human samples on Cray XE6 supercomputer in 50 h with GATK variant calling pipeline. Halvade (Decap *et al.*, 2015) is reported to process variant calling with BWA-aln alignment on the 1.5 billion paired-end reads in 159 min with 360 cores @ 2.4 GHz on 15 nodes. ViraPipe aligned 768 human samples (2.1 billion 101 bp paired-end reads) in 150 min with 1288 cores @ 2.6 GHz on 23 computing nodes. It should be noted that BWA-MEM algorithm is computationally more complex than BWA-aln and is thus slower.

5 Conclusions

To tackle the challenge of discovering viruses from thousands of samples in tolerable time with existing tools, we propose ViraPipe, a parallel pipeline for performing scalable viral metagenome analysis with Apache Spark on a computing cluster. ViraPipe integrates widely used genomics tools such as BWA aligner, MegaHit *De Novo* assembler as well as BLAST and HMMER3 search tools in flexible and reusable way. The experiments were run for assembling viral contigs, searching viral sequences with BLAST, and profiling sequences with HMMER3 in parallel from 768 human samples in 210 min with Apache Spark on Hadoop computing cluster including 1288 cores on 23 worker nodes. The results show 11× speedup on 23 computing nodes compared to sequential pipeline run on single node. ViraPipe is capable to scale from tens of gigabytes to terabytes of NGS data comprising thousands of human samples by harnessing more computing nodes to the cluster. In addition, the pipeline includes functionalities to search and manipulate NGS data in FASTA, FASTQ, SAM/BAM and BLAST output formats in parallel. The solutions in our pipeline are potentially applicable for any purposes where large-scale read alignment, *De Novo* assembly or BLAST sequence search is needed. Apache Hadoop and Spark based architecture brings a great flexibility for harnessing ViraPipe to cloud computing environment with the needed computing resources on demand.

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022

Acknowledgements

We want to thank Jim Dowling from KTH Royal Institute of Technology for maintaining the computation cluster and helping us with technical problems. We would also like to thank the peer reviewers for clarifying comments and valuable suggestions to improve the article.

Funding

This work was supported by the Nordic Information for Action eScience Center (NIASC), Swedish strategic research foundation excellence project in biomarker research (BRIGHT), Academy of Finland project Advanced Parallel Testing and Verification Methods for Distributed Systems (APTV), and Helsinki Institute for Information Technology program Foundations of Computational Health (FCHealth).

Conflict of Interest: none declared.

References

Abu-Doleh,A. and Çatalyürek,V. (2015) Spaler: Spark and graphx based de novo genome assembler. In: *2015 IEEE International Conference on Big Data (Big Data)*, pp. 1013–1018.

Arroyo Mühr,L.S. *et al*. (2015a) Does human papillomavirus-negative condylomata exist? *Virology*, **485**, 283–288.

Arroyo Mühr,L.S. *et al*. (2015b) Human papillomavirus type 197 is commonly present in skin tumors. *Int. J. Cancer*, **136**, 2546–2555.

Arroyo Mühr,L.S. *et al*. (2017) Viruses in case series of tumors: consistent presence in different cancers in the same subject. *PLoS One*, **12**, e0172308.

Brown,C. *et al*. (2012) A reference-free algorithm for computational normalization of shotgun sequencing data. arXiv:1203.4802v2 [q-bio.GN].

Bzhalava,D. *et al*. (2012) Phylogenetically diverse TT virus viremia among pregnant women. *Virology*, **432**, 427–434.

Bzhalava,D. *et al*. (2013) Unbiased approach for virus detection in skin lesions. *PLoS One*, **8**, e65953.

Bzhalava,D. *et al*. (2014) Deep sequencing extends the diversity of human papillomaviruses in human skin. *Sci. Rep*., **4**, 5807.

Chang,Y.-J. *et al*. (2012) A de novo next generation genomic sequence assembler based on string graph and mapreduce cloud computing framework. *BMC Genomics*, **13**, S28.

Decap,D. *et al*. (2015) Halvade: scalable sequence analysis with mapreduce. *Bioinformatics*, **31**, 2482–2488.

Eddy,S.R. and Pearson,W.R. (2011) Accelerated profile hmm searches. *PLOS Comput. Biol*., **7**, 1–16.

Gonzalez-Domínguez,J. *et al*. (2016) Parallel and scalable short-read alignment on multi-core clusters using upc ++. *PloS One*, **11**, e0145490.

Hall,A.B.B. *et al*. (2017) Human genetic variation and the gut microbiome in disease. *Nat. Rev. Genet*., doi:10.1038/nrg.2017.63.

Jeffrey,D. and Sanjay,G. (2004). Mapreduce: Simplified data processing on large clusters. In: *Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation*, pp. 10–10.

Li,D. *et al*. (2016) MEGAHIT v1.0: a fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods*, **102**, 3–11.

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

Li,H. *et al*. (2009) The sequence alignment/map format and samtools. *Bioinformatics*, **25**, 2078–2079.

Lin,Y. *et al*. (2011) Comparative studies of de novo assembly tools for next-generation sequencing technologies. *Bioinformatics*, **27**, 2031.

McKenna,A. *et al*. (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res*., **20**, 1297–1303.

Mistry,J. *et al*. (2013) Challenges in homology search: HMMER3 and convergent evolution of coiled-coil regions. *Nucleic Acids Res*., **41**, e121.

Mitchell,A. *et al*. (2016) Ebi metagenomics in 2016 – an expanding and evolving resource for the analysis and archiving of metagenomic data. *Nucleic Acids Res*., **44**, D595.

Niemenmaa,M. *et al*. (2012) Hadoop-bam: directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, **28**, 876.

O'keefe,S. and Greer,J. (2011) Microbial induction of immunity, inflammation, and cancer. *Front. Physiol*., **1**, 168.

Pireddu,L. *et al*. (2011) Seal: a distributed short read mapping and duplicate removal tool. *Bioinformatics*, **27**, 2159–2160.

Puckelwartz,M.J. *et al*. (2014) Supercomputing for the parallelization of whole genome analysis. *Bioinformatics*, **30**, 1508–1513.

Robinson,C.M. and Pfeiffer,J.K. (2014) Viruses and the microbiota. *Annu. Rev. Virol*., **1**, 55–69.

Rogers,G.B. and Bruce,K.D. (2012) Exploring the parallel development of microbial systems in neonates with cystic fibrosis. *MBio*, **3**, e00408–e00412.

Schumacher,A. *et al*. (2014) Seqpig: simple and scalable scripting for large sequencing data sets in hadoop. *Bioinformatics*, **30**, 119–110.1093.

Skewes-Cox,P. *et al*. (2014) Profile hidden Markov models for the detection of viruses within metagenomic sequence data. *PLoS One*, **9**, e105067.

Smelov,V. *et al*. (2016) Detection of DNA viruses in prostate cancer. *Sci. Rep*., **6**, 25235.

Stephens,Z.D. *et al*. (2015) Big data: astronomical or genomical? *PloS Biol*., **13**, e1002195.

Thomas,T. *et al*. (2012) Metagenomics – a guide from sampling to data analysis. *Microb. Inform. Exp*., **2**, 3.

Wang,J. and Mu,Q. (2003) Soap-HT-BLAST: high throughput BLAST based on Web services. *Bioinformatics*, **19**, 1863–1864.

Wylie,K.M. *et al*. (2012a) Emerging view of the human virome. *Transl. Res*., **160**, 283–290.

Wylie,K.M. *et al*. (2012b) Sequence analysis of the human virome in febrile and afebrile children. *PLoS One*, **7**, e27735.

Zaharia,M. *et al*. (2010). Spark: Cluster computing with working sets. In: *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pp. 10–10.

Zaharia,M. *et al*. (2012) Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pp. 2–2.

Zhou,W. *et al*. (2017) Metaspark: a spark-based distributed processing tool to recruit metagenomic reads to reference genomes. *Bioinformatics*, **33**, 1090.

Downloaded from https://academic.oup.com/bioinformatics/article/34/6/928/4587582 by guest on 21 August 2022