Bioinformatics, 35, 2019, i13–i22
doi: 10.1093/bioinformatics/btz356
ISMB/ECCB 2019

Large scale microbiome profiling in the cloud

Camilo Valdes[1], Vitalii Stebliankin[1] and Giri Narasimhan[1,2,*]

[1]Bioinformatics Research Group (BioRG), School of Computing and Information Sciences and [2]Biomolecular Sciences Institute, Florida International University, Miami, FL 33199, USA

*To whom correspondence should be addressed.

Abstract

Motivation: Bacterial metagenomics profiling for metagenomic whole sequencing (mWGS) usually starts by aligning sequencing reads to a collection of reference genomes. Current profiling tools are designed to work against a small representative collection of genomes, and do not scale very well to larger reference genome collections. However, large reference genome collections are capable of providing a more complete and accurate profile of the bacterial population in a metagenomics dataset. In this paper, we discuss a scalable, efficient and affordable approach to this problem, bringing big data solutions within the reach of laboratories with modest resources.

Results: We developed Flint, a metagenomics profiling pipeline that is built on top of the Apache Spark framework, and is designed for fast real-time profiling of metagenomic samples against a large collection of reference genomes. Flint takes advantage of Spark's built-in parallelism and streaming engine architecture to quickly map reads against a large (170 GB) reference collection of 43 552 bacterial genomes from Ensembl. Flint runs on Amazon's Elastic MapReduce service, and is able to profile 1 million Illumina paired-end reads against over 40 K genomes on 64 machines in 67 s—an order of magnitude faster than the state of the art, while using a much larger reference collection. Streaming the sequencing reads allows this approach to sustain mapping rates of 55 million reads per hour, at an hourly cluster cost of $8.00 USD, while avoiding the necessity of storing large quantities of intermediate alignments.

Availability and implementation: Flint is open source software, available under the MIT License (MIT). Source code is available at https://github.com/camilo-v/flint.

Contact: giri@cs.fiu.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction and background

Microbes are ubiquitous and a microbiome is a collection of microbes that inhabit a particular environmental niche such as the human body, earth soil and the water in oceans and lakes. Metagenomics is the study of the combined genetic material found in microbiome samples, and it serves as an instrument for studying microbial biodiversities and their relationships to humans. Profiling a microbiome is a critical task that tells us what microorganisms are present, and in what proportions; this is particularly important as many human diseases are linked to changes in human microbiome composition (Haiser *et al.*, 2013; Koeth *et al.*, 2013; Wu and Lewis, 2013; Zhang *et al.*, 2015), and large research projects have started to investigate the relationships between the two (The Integrative HMP iHMP Research Network Consortium, 2014).

A powerful tool for profiling microbiomes is high-throughput DNA sequencing (Metzker, 2010), and whole metagenome sequencing experiments generate data that give us a lens through which we can study and profile microbiomes at a higher resolution than 16S amplicon-based sequencing analyses (Ranjan *et al.*, 2016).

Advances in sequencing technologies have steadily reduced the cost of sequencing and have led to an ever increasing number of extremely large and complex metagenomic datasets (Ansorge, 2009; Caporaso *et al.*, 2012). The resulting computational challenge is the production of even larger intermediate results, and need for large indexes of the reference genome collections (Vernikos *et al.*, 2015), making it impossible to process on commodity workstations or laptops. Powerful multi-user servers and clusters are an option, but the cost of higher processor speeds, greater storage volumes and huge memory sizes are out of reach for small laboratories.

To deal with the barrage of sequencing data, distributed cloud computing platforms and frameworks such as Amazon Web Services (Amazon.com Inc., Amazon Web Services, 2018), Apache Hadoop (Apache Hadoop, 2018) and Apache Spark (Apache Spark, 2018) have been used by researchers by taking advantage of parallel computation and economies of scale: large sequencing workloads

are distributed in a cloud cluster that is comprised of many cheap, off-the-shelf compute nodes. These cloud-based solutions have been successfully used for human genomics (Langmead *et al.*, 2009a), transcriptomics (Roberts *et al.*, 2013) and more recently for metagenomics applications (Huang *et al.*, 2018; Zhou *et al.*, 2017).

Standard genomics and transcriptomics analyses for sequencing datasets usually begin by aligning sequencing reads to a reference genome (Trapnell and Salzberg, 2009; Wang *et al.*, 2009), and producing abundance counts (Trapnell *et al.*, 2010); but in metagenomic analyses, the alignment step is performed against a collection of reference genomes that can be extremely large, slowing down the entire operation. The MapReduce model (Dean and Ghemawat, 2008) along with the Spark framework have been popular in speeding up these crucial steps in the analysis of single-organism sequencing datasets, as researchers have framed the read-alignment and quantification tasks in terms of *map* and *reduce* operations: Langmead *et al.* used it to align human sequencing reads using the Bowtie read-mapping utility (Langmead *et al.*, 2009b) and searching for single nucleotide polymorphisms (SNPs); while Roberts *et al.* (2013) used it to speed up the quantification of human gene transcripts by the expectation-maximization (EM) algorithm.

2 Approach

2.1 Spark and MapReduce

The MapReduce model was originally developed by Google (Dean and Ghemawat, 2008), and most notably popularized by the Apache Hadoop (2018) open-source project from the Apache foundation (The Apache Software Foundation, 2018). The Apache Spark (2018) project further expanded the Hadoop project, and introduced new optimizations for calculation speeds, and programming paradigms (Zaharia *et al.*, 2012). The MapReduce model abstracts away much of the boiler-plate programming details of developing distributable applications, and frees scientists and developers to focus their work on other critical, domain-specific, areas. The model is composed of two distinct steps: the *map()* step, and the *reduce()* step. Hadoop and Spark offer basic functions that can be used as the building blocks of a distributed computing model: the *map()* function takes as input a pair of parameters that make up a tuple consisting of a key and a value; while the *reduce()* function merges the output of the *map()* function by coalescing tuples with the same key.

The MapReduce model, and the Spark framework in particular, have been employed in many DNA sequencing workflows for a number of years now (Cattaneo *et al.*, 2016; Guo *et al.*, 2018). The Crossbow project (Langmead *et al.*, 2009a) from 2009 used Spark's MapReduce implementation to identify Single Nucleotide Polymorphisms (SNPs) in human samples; eXpress-D (Roberts *et al.*, 2013) also used Spark to implement the expectation maximization (EM) algorithm for ambiguous DNA-fragment assignment. Spark has also been used in metagenomic analyses (Guo *et al.*, 2018) for mapping sequencing reads against small reference databases and for clustering metagenomes (Rasheed and Rangwala, 2013).

A natural approach to use the Spark framework for the analysis of mWGS datasets is to partition the input of reads into smaller subsets of reads to be processed by worker nodes in a Spark cluster. This strategy works well when the dataset of reads is large. The limitation of this strategy is that it does not scale to large collections of reference genomes because a data structure (index) of the reference collection of genomes must either be duplicated in each of the worker nodes, or multiple passes of the input can be used. Indexes built from large reference collections using a k-mer based strategy are often too large to be accommodated on a single commodity machine on the cloud (Nasko *et al.*, 2018). Fast k-mer based profiling strategies have been used for profiling of mWGS datasets (Schaeffer *et al.*, 2015; Wood and Salzberg, 2014). But they trade-off speed for enormous indexes. More recently, alternative index-building strategies have been developed to allow the use of large collections of references with k-mer based tools, albeit only at species-level resolutions (Zhou *et al.*, 2018), but were not designed for use with a cloud-based infrastructure.

Zhou *et al.* developed MetaSpark (Zhou *et al.*, 2017) to align metagenomic reads to reference genomes. The tool employs Spark's Resilient Distributed Dataset (RDD) (Zaharia *et al.*, 2012)—the main programming abstraction for working with large datasets—to cache reference genome and read information across worker nodes in the cluster. By using Spark's RDD, MetaSpark is able to align more reads than previous tools. MetaSpark was developed with two reference datasets of bacterial genomes: a 0.6 GB reference, and the larger 1.3 GB from RefSeq's bacterial repository. These reference sets are small compared to the 170 GB reference set of Ensembl, and because of MetaSpark's use of an RDD to hold its index, it is unlikely that MetaSpark can scale to use them: the contents of an RDD are limited to available memory, and large reference sets would require correspondingly large memory allocations. It is worth pointing out the RDD memory limitations of MetaSpark in aligning reads: it took 201 min (3.35 h) to align 1 million reads to the small 0.6 GB reference using 10 nodes (Zhou *et al.*, 2017).

SparkHit (Huang *et al.*, 2018) was developed by Huang *et al.* as a toolbox for scalable genomic analysis and also included the necessary optimizations for the preprocessing. SparkHit includes a metagenomic mapping utility called 'SparkHit-recruiter' that performs much faster than MetaSpark with similar sets of reference genomes. SparkHit performs well with large dataset of reads and small reference genome sets—the authors profiled 2.3 TB of whole genome sequencing reads against only 21 genomes in a little over an hour and a half. The limitation of SparkHit is that it builds its reference index using a k-mer strategy that does not scale to large collections of reference genomes (Nasko *et al.*, 2018), assuming that the reference database will change with each study that is analyzed. This assumption, and the method of index building, makes SparkHit unsuitable for profiling large metagenomic datasets against large collections of reference genomes.

2.2 Streaming techniques

In order to process the large quantities of both input metagenomic datasets, and the large collections of reference genomes to profile against, new analysis paradigms are required that take advantage of highly parallelizable cloud infrastructure, as well as real-time data streams for consuming large input datasets.

LiveKraken (Tausch *et al.*, 2018) was developed as a real-time classification tool that improves overall analysis times, and is based on the popular Kraken (Wood and Salzberg, 2014) method for profiling metagenomic samples in Kraken-based workflows. LiveKraken uses the same approach as the HiLive (Lindner *et al.*, 2017) real-time mapper for Illumina reads, but extends it to metagenomic datasets. LiveKraken can ingest reads directly from the sequencing instrument in illumina's binary basecall format (BCL) before the instrument's run finishes, allowing real-time profiling of metagenomic datasets. Reads are consumed as they are produced at the instrument, and the metagenomic profile produced by LiveKraken is continuously updated. LiveKraken points the way to future classification systems that use streams of data as input, but its

limitation is that it uses a k-mer based reference index—in its publication, LiveKraken was tested with an archived version of RefSeq (circa 2015) that only contained 2787 bacterial genomes. Since then, RefSeq has grown to over 50k genomes in the latest release (version 92), and creating a K-mer based index of it would require substantial computational resources.

More recently, a Spark streaming-based aligner has been developed that uses streams of data to map reads single reference genomes. The tool, StreamAligner (Rathee and Kashyap, 2018), is implemented with Spark and the Spark-streaming API, and uses novel MapReduce-based techniques to align reads to the reference genome of a single organism. Unlike other methods, it creates its own reference genome index using suffix arrays in a distributed manner that reduces index-build times, and can then be stored in memory during an analysis run. By using the Spark streaming API, StreamAligner can continuously align reads to a single reference genome without the need of storing the input reads in local storage, and although StreamAligner has high performance when using a single genome, there is no evidence if it can scale to metagenomic workflows where tens of thousands of genomes are used, and the footprint of the reference genomes are much larger than could be fit in memory.

3 Materials and methods

A natural approach to using MapReduce for large metagenomic analyses tasks is as follows. The *map* step divides the task of mapping the reads against a genomic index and the *reduce* step collects all the hits to each genome and constructs the microbial profile of the metagenomic sample. This approach works well when the same copy of the full genomic index can be farmed out to each node in the cluster. The approach fails when the index is too large to be provided to each cluster node or the collection of reads is too large for each cluster node. Streaming the reads allows for arbitrarily large collections of reads to be processed by each cluster node. Building an index of a 'shard' of the reference genome database and providing each cluster node with a smaller index allows for much larger reference databases to be used for mapping the reads (Fig. 1).

Our computational framework is primarily implemented using the *MapReduce* model (Dean and Ghemawat, 2008), and deployed in a cluster launched using the *Elastic Map Reduce* (EMR) service offered by AWS (Amazon Web Services) (Amazon.com Inc., Amazon Web Services, 2018). The cluster consists of multiple 'commodity' worker machines (a computational 'worker' *node*), each with 15 GB of RAM, 8 vCPUs (each being a hyperthread of a single Intel Xeon core) and 100 GB of disk storage. Each of the worker computational nodes will work in parallel to align the input sequencing DNA reads to a 'shard' of the reference database (Fig. 2); after the alignment step is completed, each worker node acts as a regular Spark executor node. By leveraging the work of multiple machines working at the same time, FLINT is able to align a large number of reads to a large database of reference genomes in a much more efficient manner than that achieved by using a single powerful machine.

3.1 Cluster provisioning

A Spark (Apache Spark, 2018) cluster was created using the AWS Console with the following software configuration: EMR-5.7.0, Hadoop 2.8.4, Ganglia 3.7.2, Hive 2.3.3, Hue 4.2.0, Spark 2.3.1 and Pig 0.17.0 in the US East (N. Virginia) region.

The cluster is composed of homogeneous machines for both the driver node and worker nodes, and each machine is an Amazon machine instance of type c4.2xlarge. These instances contain 8 vCPUs, 15 GB of RAM, 100 GB of EBS storage and each cost on average $0.123 USD to run per hour on the 'us-east' availability zone on the Spot (EC2 Spot Market, 2018) market as of this writing in January 2019. Newer instances (c5.2xlarge) are also available for use, but their availability is infrequent in large numbers, in addition to having a higher cost per hour to run.

Resilient Distributed Datasets (RDD) (Zaharia *et al.*, 2012) are robust programming abstractions that can be used to persist data across a cluster of machines. We ingest reads from datastreams in batches of 500 000 reads that are processed by our mapreduce pipeline. Reads are consumed either directly from their location in an Amazon S3 bucket, or from a datastream source such as a Kafka or Kinesis source. An RDD of the input read stream is created in the master node that is then broadcasted out into all the worker nodes in the cluster. The input RDD of reads is partitioned into sets of reads that are each independently aligned to a reference genome partition in each of the worker nodes.

3.2 A 'double' MapReduce

An obvious way to perform MapReduce for metagenomic analysis is to have the Map function produce tuples of the form $\langle g, 1\rangle$, for every read r that is aligned to genomes g, while the Reduce function aggregates all tuples of the form $\langle g, 1\rangle$ to obtain the abundance of genome g in the sample being analyzed, effectively generating output tuples of the form $\langle g, \mathcal{A}(g)\rangle$, where $\mathcal{A}(g)$ is the reported abundance of genome g in the sample being analyzed.

Unfortunately, a read may align to multiple genomes. Instead of counting a hit for every genome that the read aligns to, or counting it for only one of the genomes that the read aligns to, we follow the algorithm of Valdes *et al.* (2015), which assigns fractional counts for the genomes that a read aligns to. In order to implement this, we employ a novel double MapReduce steps, thus making it a multistage operation. In the modified MapReduce, the Map function generates alignments in SAM format (Li and 1000 Genome Project Data Processing Subgroup, 2009) by dispatching a subprocess of the Bowtie2 aligner and produces tuples of the form $\langle r, (g, 1)\rangle$, for every read r that is aligned to genomes g. All tuples for the same read are aggregated by the first Reduce step to generate tuples of the form $\langle r, (g, 1/\mathcal{C}(r))\rangle$. The second Map step generates contributions of reads for a given genome, and the second Reduce step aggregates all tuples of the form $\langle g, c\rangle$ to obtain the abundance of genome g in the sample being analyzed, effectively generating output tuples of the form $\langle g, \mathcal{A}(g)\rangle$, where $\mathcal{A}(g)$ is the reported abundance of genome g in the sample being analyzed obtained by aggregating all the fractional contributions of reads that map to that genome. Note that all intermediate tuples are stored in RDDs, one for each step.

3.3 Reference genome preparation

Before we can use the bacterial genomes in the cluster, they need to be prepared. The process entails creating a Bowtie2 index for each shard of the reference database, and specific details on this procedure can be found in Section 2.1 of the supplementary manuscript. Briefly, the reference genomes are divided into smaller partitions that are each independently indexed by Bowtie2. The index preparation step can take considerable computational resources and time with a single machine. A parallel version of the indexing system can greatly improve performance and will be completed in the next release of FLINT. Once the partitions have been indexed they are then

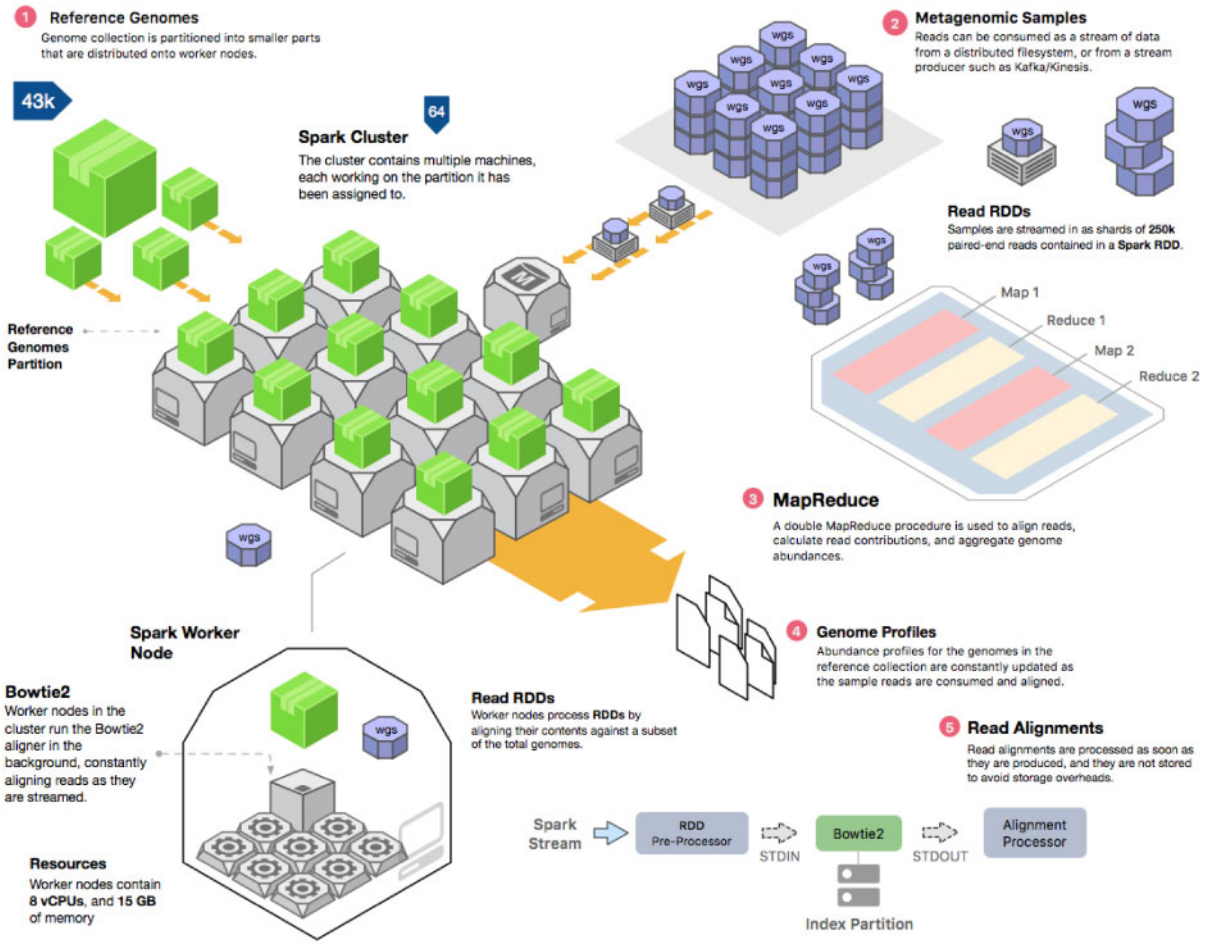

Fig. 1. Overview of the Flint System. Reference genomes are partitioned so that a large reference set is be distributed across a Spark cluster, and the number of partitions matches the number of worker nodes. Samples are streamed into the cluster to avoid storage overheads as shards of 250k reads. Reads are aligned to the distributed reference genomes using a double MapReduce pipeline that continually updates metagenomic profiles as samples are streamed into the cluster. Read alignments are never stored, and are processed by each worker node as soon as they are produced

copied to an Amazon S3 (2018) bucket that serves as a staging location for the reference shards. The staging S3 bucket holds the index so that worker nodes can copy it during their provisioning step and the analysis can start; the S3 bucket is also public, and researchers can download copies of the prepared indices for their use.

It should be noted that Ensembl's bacterial genome collections have grown only modestly in the last couple of releases to minimize redundancy, and reference indices for new Ensembl releases can be built relatively quickly with utility scripts provided by Flint. The cost of building a partitioned reference index is only accrued the first time it is built for a cluster of a particular size, and as part of the release of the Flint project, we are making available partitioned indices of Ensembl (v.41) of sizes 48, 64, 128, 256 and 512 which should be useful for researchers employing clusters of those sizes. These indices, along with the scripts necessary to build future versions, can be found at the GitHub repository.

We currently use minimal annotations that keep track of basic attributes for each bacterial strain; these include taxonomic identifiers, assembly lengths, etc. Future releases of the software will include a more robust annotations package that will contain data on gram staining, pathogenicity and other properties.

Flint uses a streaming model to quickly map a large number of reads to a large collection of reference bacterial genomes by using a distributed index. The Bowtie2 DNA aligner is used internally in Spark worker nodes to align reads to the local partition of the reference index, by using a MapReduce that continuously streams reads into worker nodes. Output alignments are parsed and tabulated by worker nodes, and then sent back to master node as alignment tasks finish. Flint can be deployed on any Spark cluster, as long as the necessary software dependencies are in place; the partitioned reference index for Ensembl's 43k genomes is made available at the Flint website, and scripts are provided as part of the provisioning step that copy the partitions into worker nodes.

4 Results and discussion

4.1 Comparison to existing tools

Flint was evaluated by comparing abundance profiles generated with Flint to those provided by HMP and those generated by Kraken (Wood and Salzberg, 2014). Note that Kraken is a *k*-mer based algorithm to align reads to genomic sequences and is known to be one of the most accurate ones (McIntyre *et al.*, 2017).

We selected an anterior nares sample (SRS019067) with 528k reads from the Human Microbiome Project (HMP) and analyzed it with Kraken (2.0.7-beta) and Flint and compared the results to those provided by HMP in their community abundance profiles. HMP reported 36.7% aligned reads using a bacterial database of 1751 genomes, while Kraken was able to classify 36% of the reads

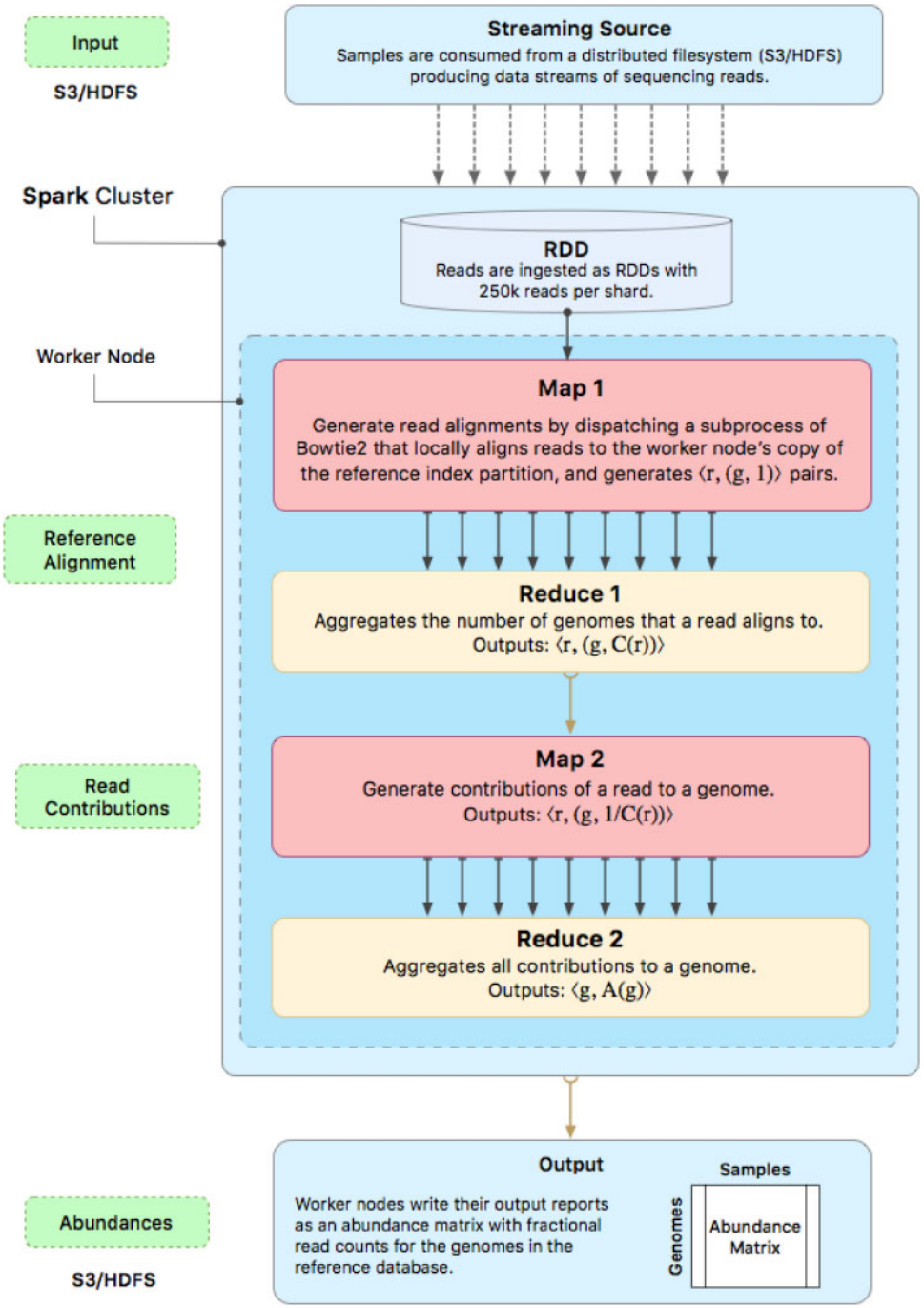

Fig. 2. MapReduce workflow. Metagenomic samples can be streamed in from a distributed filesystem into the cluster were they are stored in an RDD. The first Map step generates alignments through Bowtie2 and feeds its resulting pairs to the first Reduce step, which aggregates the genomes that a single reads aligns to. The second Map step generates read contributions that are used in the second Reduce step to aggregate all the read contributions for a single genome. An output abundance matrix is generated which contains the abundances for each genome

using their RefSeq bacterial database of 14 506 genomes; in contrast, FLINT was able to align 81% of the reads using Ensembl's 43k bacterial genomes. The increase number of aligned reads is due to the larger number of genomes in Ensembl—Kraken uses RefSeq's so-called 'complete' bacterial genomes, while Ensembl contains many draft genomes that increases the probability for mapping a read. FLINT also aligns reads with Bowtie2 directly to the bacterial strain genomes, and does not apply lowest common ancestor (LCA) assignment to reads as Kraken does, which should mitigate any database diversity influences (genus, species and strain ratios) as noted by Nasko *et al.* (Nasko *et al.*, 2018). As shown in Supplementary Figure S4, both FLINT and Kraken identify roughly the same set of genera, but at the species level, FLINT identifies significantly more species.

MetaSpark (Zhou *et al.*, 2017) and SparkHit (Huang *et al.*, 2018) are spark-based methods with a cluster infrastructure similar to FLINT but their lack of support for large genome references makes direct comparison impossible. MetaSpark has a 201 min runtime for

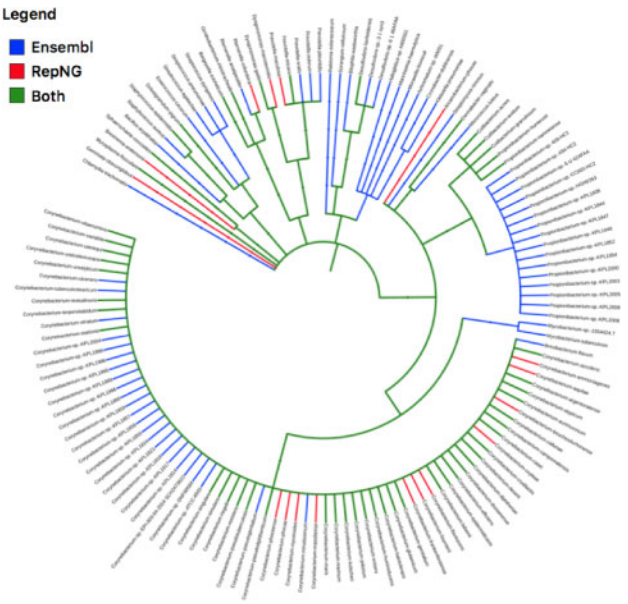

Fig. 3. Phylogenetic tree of taxa identified by Flint using 43k Ensembl bacterial genomes (blue), and 5k NCBI's Genomes references (red) with an input of 1 M randomly selected reads from the HMP anterior nares sample (SRS015996). Genomes are identified if the average coverage in their genomic sequence is 80% or more

1 million reads with 10 nodes, profiled against a 0.6 GB reference of bacterial genomes from NCBI. In comparison, FLINT takes 67 s to profile 1 million paired-end reads against Ensembl's 43 552 genomes (170 GB) with 64 nodes.

4.2 Reference genome collections

To test the speed of our read alignment step, we downloaded a reference collection of bacterial genomes from the Ensembl Bacteria (2018) repository (version 41). A total of 43 552 bacterial genomes (strain level) were downloaded in FASTA format, accounting for 4.6 million individual FASTA assembly references. The collection included reference sequences for fully assembled chromosomes and plasmids, as well as containing sequences for draft-quality supercontigs, the latter accounting for most of the reference files in the database. The Ensembl bacterial genomes (v.41) were downloaded from the public FTP site at ftp.ensemblgenomes.org. Ensembl stores the FASTA files in 'collection' directories, and we recursively downloaded the 'dna' directory in each of the bacterial sub-folders. In total, 4 672 683 FASTA files were downloaded, with a data footprint on disk of just over 170 GB, accounting for 43 552 bacterial strains.

Creating the Bowtie2 index for the bacterial genomes is a one-time operation as the index can be reused across cluster deployments. With a 64 worker-node cluster, we created 64 reference shards, each having a size of 2.6 GB on average. The total sequential indexing time for the 64 shards was 1d 20 h 4 m 33 s on a single machine, but we also used an LSF cluster (IBM Spectrum LSF., 2019) that indexed the 64 shards in parallel, and brought down the total indexing time to just over 3 h.

Existing metagenomic profiling tools such as MetaSpark and SparkHit use an archived version of RefSeq as their reference genomes database—MetaSpark's RefSeq bacterial references was for 1.3 GB of size. Given the fact that the Ensembl database used by FLINT is roughly ten times larger, we looked into how a metagenomic profile could be different by looking at how many genomes are identified by using a large or small reference collection. To do this we randomly selected 1 M reads from an HMP anterior nares sample (SRS015996) and aligned its reads using Bowtie2 to two genome reference indices: the large collection created from the 43k Ensembl bacterial genomes, and the small collection created from 5591 bacterial representative and reference genomes from NCBI's Genomes (RepNG). We investigated how many clades are identified by both references, and Figure 3 displays the results. Figure 3 shows a phylogenetic tree [created with the Interactive Tree Of Life (iTOL) visualization tool (Letunic and Bork, 2016)] showing the differences in the phylogenetic diversity of the taxa identified in the anterior nares sample. Genomes are called as 'present' by selecting only those genomes that have an average coverage greater than 80% along their genomic sequence. Nodes at the inner level of the figure represent the phylum taxonomic level, while nodes in the outer rings are at the species level. Green branches represents the clades identified by both references, blue branches represent clades identified by Ensembl, and red branches are clades identified by the RepNG reference set. Note that the number of clades identified by Ensembl at the higher Class and Genus taxonomic levels outnumber those identified when only using the RepNG subset.

4.3 Experimental setup

As mentioned earlier, the computational framework is primarily implemented using the *MapReduce* model (Dean and Ghemawat, 2008), and deployed in a cluster launched in Amazon Web Services (Amazon.com Inc., Amazon Web Services, 2018) *Elastic Map Reduce* (EMR) service. The cluster consists of multiple worker machines (i.e. a computational 'worker' *node*), each with 15 GB of RAM, 8 vCPUs (each being a hyperthread of a single Intel Xeon core) and 100 GB of disk storage. Each of the worker computational nodes will work in parallel to align the input sequencing DNA reads to a shard of the reference database; after the alignment step is completed, each worker node acts as a regular Spark executor node. By leveraging the work of multiple machines working at the same time, we are able to align millions of reads to the over 43k reference genomes in a much more efficient manner than either using only a single machine with considerable computational resources, or using other parallel computation approaches. Benchmarking tests were performed in Spark clusters of size 48, 64 and 128 worker nodes, all deployed in Amazon's EMR service for very low costs.

4.4 Measuring accuracy using simulated datasets

To get a measure of the accuracy of FLINT's read-alignment pipeline, and to test the robustness of the streaming infrastructure, we simulated synthetic Illumina reads using the InSilicoSeq (Gourlé *et al.*, 2018) metagenomic simulator. We created three replicate dataset groups to test the accuracy of the overall pipeline, and to verify that the streaming system would not introduce any duplicate artifacts, or that the reduce steps in the Spark cluster would not exclude any of the output alignments. Each replicate group consists of 12 datasets ranging from a dataset with 1 read to a dataset with 1 million reads, created with a log-normal abundance profile, and using the default error model for the HiSeq sequencing instrument available in InSilicoSeq. Specific details on the simulation protocol, cluster configuration and detailed results for each replicate set are available in the Supplementary Materials.

Table 1 outlines the results for the synthetic HiSeq datasets. Dataset evaluations were performed on a 64 worker-node cluster in AWS, with each worker node containing 8 vCPUs and 15 GB of

Table 1. HiSeq synthetic datasets

Reads	Alignments	Time	Alignment rate (%)	% Sensitivity
1	1	2 s 344 ms	100	100
10	23	2 s 400 ms	100	100
100	172	2 s 376 ms	100	100
1000	1356	2 s 455 ms	100	100
5000	8592	2 s 517 ms	90	98
10 000	23 791	3 s 193 ms	94	99
50 000	74 543	5 s 138 ms	96	100
100 000	103 835	8 s 320 ms	93	99
250 000	187 349	15 s 788 ms	95	100
500 000	275 917	29 s 18 ms	93	97
750 000	513 954	45 s 91 ms	95	99
1M	617 933	1 m 14 s 713 ms	96	99

Note: Average alignment times and alignment rates for three synthetic datasets aligned against Ensembl's 43k bacterial genomes. Sensitivity is the proportion of paired-end reads that were mapped correctly to the genome from which they were generated. Evaluations were performed on a 64 worker-node Spark cluster.

Table 2. Initial cluster benchmarks

Paired-end reads	Alignments	Time (ms)	Memory (GB)
1	0	2 s 320 ms	4
10	36	2 s 422 ms	4
100	902	2 s 336 ms	4
1000	9252	2 s 316 ms	4.3
5000	53 918	2 s 455 ms	4.5
10 000	106 160	2 s 700 ms	4.9
50 000	538 594	5 s 437 ms	5.2
100 000	1 006 122	8 s 318 ms	5.8
250 000	2 349 518	17 s 164 ms	6.4
500 000	5 327 040	33 s 950 ms	7.6
750 000	8 439 356	50 s 880 ms	9.5
1M	10 710 420	1 m 7 s 609 ms	10.3

Note: Average alignment times in a 64 worker-node cluster for a set of randomly selected reads from a HMP anterior nares sample. The number of alignments column contains the output alignments that are generated by each set of reads; these alignments are processed as soon as they are produced and are not stored, therefore minimizing the local storage requirements necessary for profiling metagenomic samples.

memory. FLINT achieves good performance with the HiSeq dataset achieving 99% sensitivity across all three HiSeq replicates. Alignment times on the 64 node Spark cluster using the database of over 43k Ensembl bacterial genomes show that 1 million reads are aligned in just over 1 min with no loss of sensitivity. The 'Alignments' column contains the number of alignments that are produced as output for each dataset—these output alignments are not stored by the system, but rather they are processed as soon as they are generated by the worker nodes in the cluster.

4.5 Human metagenomic samples

After verifying the performance of the FLINT system on simulated datasets, we tested the capabilities of the system on real metagenomic samples from the Human Microbiome Project (HMP) (Human Microbiome Project Consortium, 2012), which was generated using an Illumina-based sequencing system. We therefore expected a comparable performance with the HMP data as with the synthetic dataset.

4.6 Cluster benchmarks

Before testing the system with full human metagenomic samples, we ran a benchmark of randomly sampled paired-end reads from a HMP anterior nares sample (SRS015996) to confirm our previous observations on the synthetic datasets. Each of these read datasets was then processed through the FLINT system running on a 64 worker-node cluster in AWS. Table 2 presents the runtimes for each of the datasets, and FLINT can process 1 million reads in about 67 s.

4.7 Full human samples

We analyzed 173 million paired-end reads from three HMP samples sequenced from anterior nares (SRS019067, 528k reads), stool (SRS065504, 116 M reads), and supragingival plaque (SRS017511, 56 M reads). These paired-end reads represent samples with varying levels of metagenomic diversity. For the purposes of analysis and the comparison of our execution pipeline, we created diversity classes defined by the number of unique genera present in each sample. To obtain our diversity classes, we analyzed 753 HMP samples for their abundance profiles and surveyed the number of unique genera as reported in the community abundance profiles provided by HMP (see Supplementary Materials for details); we then selected representative samples that contained 133 unique genera (high diversity class), 60 unique genera (medium diversity class) and 8 unique genera (low diversity class). The reasoning for using these samples was to test the performance of the FLINT system in samples with varying degrees of metagenomic diversity. We speculated that low diversity samples would contain reads from a relatively small number of organisms, and therefore the alignment system would not spend too much time finding their genomes of origin. In contrast, the high diversity samples would contain reads from a large number of organisms, and the alignment system would spend more time and resources locating their origins.

Table 3 contains the results from running the three samples through the FLINT system. The sample with the biggest number of paired-end reads, sample SRS065504 with 116 million paired-end reads, was profiled against Ensembl's 43k genomes in about 105 min. The sample with the second largest number of paired-end reads, i.e. sample SRS017511 with 56 million paired-end reads, was profiled against the 43k genomes in about 94 min; while the sample with the lowest number of paired-end reads was profiled in 53 s. Note that the sample with 116 million paired-end reads was processed in about 10 min more than the sample with 56 million paired-end reads—this sample with 56 million reads is the sample that contains the highest number of unique genera (highest metagenomic diversity, 133 versus 60 in the larger sample). Since more alignments were found, the reads required more time to be processed.

4.8 Streaming performance

The samples in Table 3 were streamed into the cluster through Spark's streaming engine. The entire sample is never ingested all at once, but rather, we stream in shards of each sample so that we do not overrun the cluster with so much data that it would cause a cluster failure. To find the ideal number of reads that we could use a size of a stream shard, we looked at the results in Table 2 and Figure 4. Figure 4B displays a logarithmic curve of the alignment times for all 12 sizes of the paired-end read datasets, and while we can align 1 million reads in about 67 s, doing so creates so many alignments that each of the Spark executor processes running in each worker node could run out of memory. We looked for the 'knee-in-the-curve' in Figure 4B, marked by the vertical magenta line, and identified a size of 250k paired-end reads as a good trade-off between

Table 3. HMP sample analysis

Diversity class	Unique genera	Sample ID	Paired-end reads	Alignment execution time	Streamed shards	Avg. alignments per stream shard
Low	8	SRS019067	528 988	0 h 0 m 53 s	2	1 763 227
Medium	60	SRS065504	116 734 970	1 h 45 m 30 s	234	1 471 036
High	133	SRS017511	56 085 526	1 h 34 m 51 s	113	1 535 626

Note: Diversity classes were established based on the number of unique genera in 753 HMP samples. Three samples were selected from each diversity class and analyzed in a 64 worker-node cluster. Alignment execution time measures the total time to align all the sample reads against Ensembl's 43k bacterial genomes. The streamed shards are the number of 250k read sets that are streamed into the cluster, and the average alignment per shard is the average number of alignments produced by each shard.

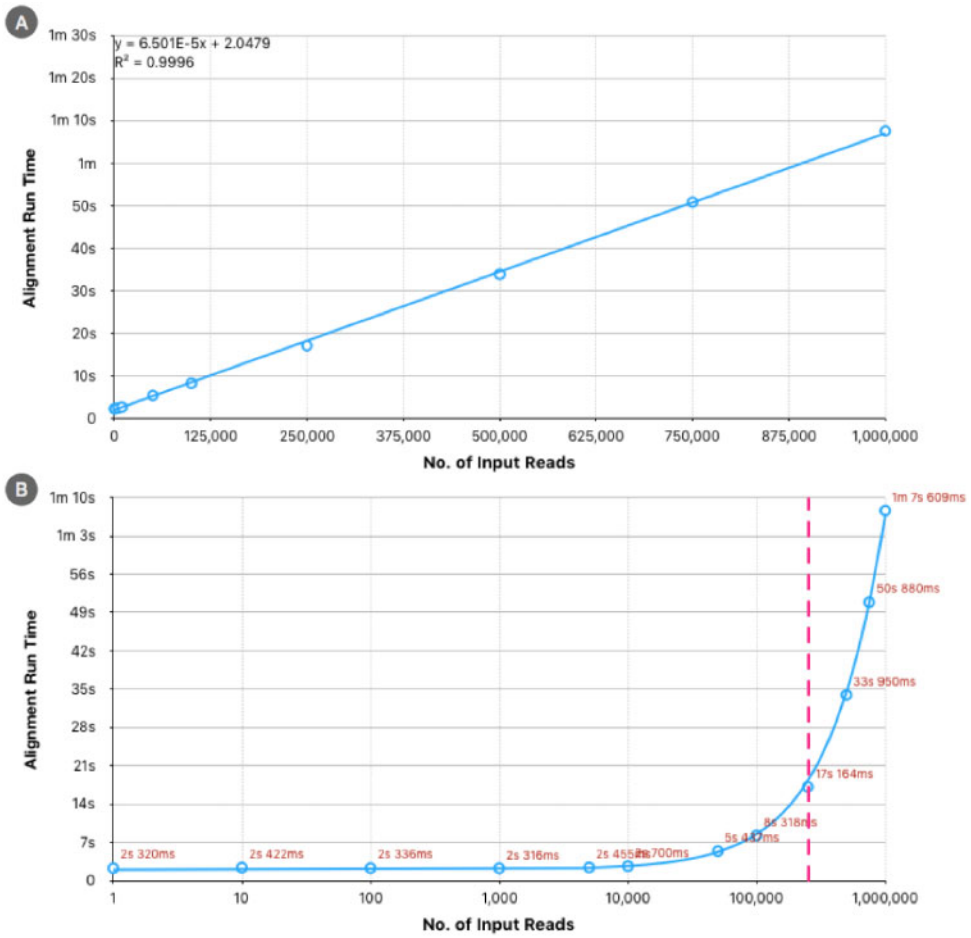

Fig. 4. Initial Benchmarks. (**A**) The running time for 12 paired-end read datasets in a 64 worker-node cluster. These 12 datasets were used to estimate the optimal number of reads that a 64 worker-node cluster could handle without any memory pressure, or network issues. Note that while 1 million paired-end reads can be mapped in 67 s against 43k bacterial strains, it is not ideal as the cluster's memory is overwhelm with alignments. (**B**) The logarithmic running time of the 12 datasets, and the 250k paired-end read dataset was chosen as a good trade-off between speed and resource availability

shard size and cluster performance. When we analyzed the three HMP samples in Table 3 we set the streaming shard size to 250k reads, and 2 shards were created for the anterior nares sample (low diversity, 500 k reads), 234 shards were created for the stool sample (medium diversity, 116 M reads) and 113 shards for the supragingival plaque sample (high diversity, 56 M reads).

4.9 Cloud costs

All experiments were conducted in Amazon's Elastic MapReduce service (EMR) (Amazon EMR, 2018) and used the 'c4.2xlarge' machine instance type. These machines contain 8 vCPUs, 15 GB of RAM and 100 GB of EBS storage; at the time of the experimental runs, each machine cost $0.123 USD in the Amazon's Spot market (EC2 Spot Market, 2018). All results reported here were obtained on a cluster of 65 total machines (64 worker-nodes, 1 master node) with a cost of $0.123 USD per node, for an overall cluster cost of $8.00 per hour.

5 Conclusion

In this work we have shown how large metagenomic samples comprising millions of paired-end reads can be profiled against a large collection of reference bacterial genomes in a fast and economical way. Our implementation relies on the freely available Spark

framework to distribute the alignment of millions of sequencing reads against Ensembl's collection of 43k bacterial genomes. The reference genomes are partitioned in order to distribute the genome sequences across worker machines, and this allows us to use large collections of reference sequences. By using the well-known Bowtie2 aligner under the hood in the worker-nodes, we are able to maintain fast alignment rates, without loss of accuracy.

To date, profiling metagenomic samples against thousands of reference genomes has not been possible for research groups with access to modest computing resources. This is due to the size of the reference genomes and the financial costs of the computing resources necessary to employ them. By using distributed frameworks such as Spark, along with affordable cloud computing services such as Amazon's EMR, we are able to distribute a large collection of reference genomes (totaling 170 GB of reference sequence, and 4.6 million assembly FASTA files) and use a MapReduce strategy to profile millions of metagenomic sequencing reads against them in a matter of hours, and at minimal financial costs, thus bringing sophisticated metagenomic analyses within reach of small research groups with modest resources.

FLINT is open source software written in Python and available under the MIT License (MIT). The source code can be obtained at the following GitHub repository: https://github.com/camilo-v/flint. The repository includes instructions and documentation on provisioning an EMR cluster, deploying the necessary partitioned reference genome indices into worker nodes, and launching an analysis job. Supplementary Materials, simulation datasets and partitioned reference indices can be found in the FLINT project website at http://biorg.cs.fiu.edu/.

Acknowledgments

The authors would like to thank Eric S. Johnson and John Flynn at the Computer Science Department's IT Support group for their help with managing the reference genomes datasets, and to the members of the Bioinformatics Research Group (BioRG) for valuable feedback on the project. Also helpful in obtaining initial results was The High Performance Group (HPC) group at FIU's Division of Information Technology.

Funding

This work was supported in part by Amazon's 'AWS Cloud Credits for Research' program, awarded to CV. The work of GN was supported by National Institute of Health (award 580 number 1R15AI128714-01), Department of Defense (contract number 581 W911NF-16-1-0494) and National Institute of Justice (award number 582 2017-NE-BX-0001).

Conflict of Interest: none declared.

References

Amazon Elastic MapReduce, EMR. (2018) https://aws.amazon.com/emr (16 November 2018, date last accessed).

Amazon Simple Storage Service, S3. (2018) https://aws.amazon.com/s3 (16 November 2018, date last accessed).

Amazon Web Services, AWS. (2018) https://aws.amazon.com/ (17 October 2018, date last accessed).

Ansorge,W.J. (2009) Next-generation DNA sequencing techniques. *New Biotechnol.*, **25**, 195–203.

Apache Hadoop. (2018) http://hadoop.apache.org (17 October 2018, date last accessed).

Apache Spark. (2018) http://spark.apache.org (17 October 2018, date last accessed).

Caporaso,J.G. *et al.* (2012) Ultra-high-throughput microbial community analysis on the Illumina HiSeq and MiSeq platforms. *ISME J.*, **6**, 1621–1624.

Cattaneo,G. *et al.* (2016) MapReduce in computational biology – a synopsis. In: Federico,R. *et al.* (eds) *Advances in Artificial Life, Evolutionary Computation, and Systems Chemistry*. Springer, Cham., pp. 53–64.

Dean,J. and Ghemawat,S. (2008) MapReduce: simplified data processing on large clusters. *Commun. ACM*, **51**, 107–113.

EC2 Spot Market. (2018) https://aws.amazon.com/ec2/spot (16 November 2018, date last accessed).

Ensembl Bacteria. (2018) https://bacteria.ensembl.org/index.html (15 October 2018, date last accessed).

Gourlé,H. *et al.* (2018) Simulating Illumina metagenomic data with InSilicoSeq. *Bioinformatics*, **35**, 521–522.

Guo,R. *et al.* (2018) Bioinformatics applications on Apache Spark. *GigaScience*, **7**, giy098.

Haiser,H.J. *et al.* (2013) Predicting and manipulating cardiac drug inactivation by the human gut Bacterium *Eggerthella lenta*. *Science (New York, NY)*, **341**, 295–298.

Huang,L. *et al.* (2018) Analyzing large scale genomic data on the cloud with Sparkhit. *Bioinformatics (Oxford, England)*, **34**, 1457–1465.

Human Microbiome Project Consortium (2012) A framework for human microbiome research. *Nature*, **486**, 215–221.

IBM Spectrum LSF. (2019) https://www.ibm.com/support/knowledgecenter/en/SSWRJV/product_welcome_spectrum_lsf.html (20 March 2019, date last accessed).

Koeth,R.A. *et al.* (2013) Intestinal microbiota metabolism of L-carnitine, a nutrient in red meat, promotes atherosclerosis. *Nat. Med.*, **19**, 576–585.

Langmead,B. *et al.* (2009a) Searching for SNPs with cloud computing. *Genome Biol.*, **10**, R134.

Langmead,B. *et al.* (2009b) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.

Letunic,I. and Bork,P. (2016) Interactive tree of life (itol) v3: an online tool for the display and annotation of phylogenetic and other trees. *Nucleic Acids Res.*, **44**, gkw290.

Li,H. *et al.* and 1000 Genome Project Data Processing Subgroup. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, **25**, 2078–2079.

Lindner,M.S. *et al.* (2017) HiLive: real-time mapping of Illumina reads while sequencing. *Bioinformatics (Oxford, England*, **33**, 917–319.

McIntyre,A.B.R. *et al.* (2017) Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. *Genome Biol.*, **18**, 182.

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

Nasko,D.J. *et al.* (2018) RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. *Genome Biol.*, **19**, 165.

Ranjan,R. *et al.* (2016) Analysis of the microbiome: advantages of whole genome shotgun versus 16S amplicon sequencing. *Biochem. Biophys. Res. Commun.*, **469**, 967–977.

Roberts,A. *et al.* (2013) Fragment assignment in the cloud with eXpress-D. *BMC Bioinformatics*, **14**, 358.

Rasheed,Z. and Rangwala,H. (2013) A map-reduce framework for clustering metagenomes. In: *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, May 20–24, 2013*. IEEE Computer Society, Washington, DC, USA, pp. 549–558.

Rathee,S. and Kashyap,A. (2018) StreamAligner: a streaming based sequence aligner on Apache Spark. *J. Big Data*, **5**, 8.

Schaeffer,L. *et al.* (2015) Pseudoalignment for metagenomic read assignment. *arXiv.org*.

Tausch,S.H. *et al.* (2018) LiveKraken – Real-time metagenomic classification of Illumina data. *Bioinformatics (Oxford, England)*.

The Apache Software Foundation. (2018) https://www.apache.org. (17 October 2018, date last accessed).

The Integrative HMP iHMP Research Network Consortium. (2014) The integrative human microbiome project: dynamic analysis of microbiome-host omics profiles during periods of human health and disease. *Cell Host Microbe*, **16**, 276–289.

Trapnell,C. *et al*. (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat. Biotechnol*., **28**, 511–515.

Trapnell,C. and Salzberg,S.L. (2009) How to map billions of short reads onto genomes. *Nat. Biotechnol*., **27**, 455–457.

Valdes,C. *et al*. (2015) Detecting bacterial genomes in a metagenomic sample using NGS reads. *Stat. Interface*, **8**, 477–494.

Vernikos,G. *et al*. (2015) Ten years of pan-genome analyses. *Curr. Opin. Microbiol*., **23**, 148–154.

Wang,Z. *et al*. (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet*., **10**, 57–63.

Wood,D.E. and Salzberg,S.L. (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol*., **15**, R46.

Wu,G.D. and Lewis,J.D. (2013) Analysis of the human gut microbiome and association with disease. *Clin. Gastroenterol. Hepatol*., **11**, 774–777.

Zaharia,M. *et al*. (2012) Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: *9th USENIX Symposium on Networked Systems Design and Implementation*, pp. 15–28.

Zhang,Y. *et al*. (2015) Metagenomics: a new way to illustrate the crosstalk between infectious diseases and host microbiome. *Int. J. Mol. Sci*., **16**, 26263–26279.

Zhou,W. *et al*. (2018) ReprDB and panDB: minimalist databases with maximal microbial representation. *Microbiome*, **6**, 15.

Zhou,W. *et al*. (2017) MetaSpark: a spark-based distributed processing tool to recruit metagenomic reads to reference genomes. *Bioinformatics (Oxford, England)*, **33**, 1090–1092.