1

CLOUD COMPUTING FOR NEXT-GENERATION SEQUENCING DATA ANALYSIS

XUAN GUO, NING YU, BING LI, AND YI PAN

Department of Computer Science, Department of Biology, Georgia State University, Atlanta, GA, USA

1.1 INTRODUCTION

Since the automated Sanger sequencing method dominated in the 1980s (1), considered as the first-generation sequencing technology, researchers first have the opportunity to construct steadily an effective ecosystem for the production and consumption of genomic information. A large number of computational tools have been developed to decode the biological information from the sequence databases in the ecosystem. Due to the expensive cost of using the first-generation sequencing technology, only a few bacteria, whose organisms possess relatively small and simple genomes, were sequenced to publish. However, along with the completion of the Human Genome Project in the beginning of the 21st century, studies on large-scale genome analysis became feasible depending on an unprecedented proliferation of genomic sequence data, which was unimaginable only a few years ago. The advent of newer methods of sequencing, known as next-generation sequencing (NGS) technologies (2), threatens the conventional genome informatics ecosystem in terms of the storage space, as well as the efficiencies of transitional tools when analyzing such huge amounts of data. The medical discoveries of the future will largely rely on our ability to dig out the "treasure" from the massive biological data. Thus,

Computational Methods for Next Generation Sequencing Data Analysis, First Edition.
Edited by Ion I. Măndoiu and Alexander Zelikovsky.
© 2016 John Wiley & Sons, Inc. Published 2016 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/Mandoiu/NextGenerationSequencing

unprecedented demands are placed on the storage and analysis approaches for big data. Moreover, voluminous data may consume all network bandwidth available to the organization and cause traffic trouble in the network because of the uploading and downloading for large data sets. In addition, local data centers will constantly suffer other issues, including control of data access, sufficient input/output, data backup, power supply, and cooling of computing resources. All of these obstacles have led to the solution in the form of cloud computing, which has become a significant technology in big data era and exerted revolutionary influences on both academy and industry.

1.2 CHALLENGES FOR NGS DATA ANALYSIS

Since the 1980s, the genomic ecosystem (Figure 1.1 (3)) for production and consumption of genomic information consists of sequencing lab, archives, power users, and casual users. The sequencing labs submitted their data to big archival databases, such as GenBank of National Center for Biotechnology Information (NCBI) (4), European Bioinformatics Institute EMBL database (5), and Sequence Read Archive (SRA, previously known as Short Read Archive) (6). Most of these databases maintain, organize, and distribute sequencing data, and also provide data access and associated tools to both power users and casual users freely. Most users obtain information either via websites created by archival databases or by value-added integrators.

The basis for the above ecosystem is Moore's law (7), which describes a long-term trend first introduced in 1965 by Intel co-founder Gordon Moore. Moore's law stated that "the number of transistors that can be placed on an integrated circuit board is increasing exponentially, with a rate of doubling in roughly 18 months" (8). The trend has remained true for approximately 40 years across multiple changes in semiconductor and manufacturing techniques. Similar phenomena have been noted for disk storage: hard drive capacity doubles roughly annually (Kryder's law) (9); and network capacity that the cost of sending a bit of information over optical networks halves every 9 months (Nielsen's law and Butter's law) (10). Along with the improvement of genome sequencing technology, the increasing rate of time for DNA sequencing was approximating the growth of computing and storage capacity at the beginning. The archival databases and computational biologists did not need to worry about running out of disk storage space or not having access to sufficiently powerful

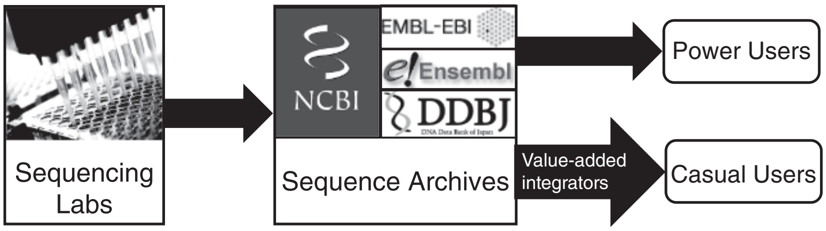

Figure 1.1 The old genome informatics ecosystem prior to the advent of next-generation sequencing technologies (3).

networks because the slight difference between two rates allowed them to upgrade their capacity ahead of the curve.

However, a deluge of biological sequence data has been generated since the Human Genome Project was completed in 2003. The advent of NGS technologies in the mid-2000s increases the slope of the DNA sequencing curve abruptly and now threatens the conventional genome informatics ecosystem. The commercially available NGS technologies, including 454 Sequencer (11), Solexa/Illumina (12), and ABI SOLiD (13), generated a tsunami of petabyte-scale genomic data, which flooded biological databases as never before. In terms of the prices of hard disk and DNA sequencing, we illustrate this by using a long-term trend (Figure 1.2) (7) plotted by Stein (14). Note that exponential curves are drawn as straight lines in the logarithmic scale. According to the figure, it is clear that the cost of storing a byte of data was halved every 14 months during 1990–2010. On the contrary, the cost of sequencing a base was halved every 19 months during 1990–2004, which is more slow than the unit cost of storage did. After the widespread use of NGS technologies, the cost of sequencing a base was halved down to every 5 months, which leads to the drop in the cost of genome sequencing several times faster than the cost of storage. It is not difficult to predict that it will cost us less to sequence a base of DNA than to store it on a hard disk sometime shortly. There is no guarantee to accelerate the trends all the time, but recently announced results by Illumina (15), Pacific Biosystems (16), Helicos (17), and Ion Torrent (18) ensure the continuing of the trends for at least another half-century. The development of NGS makes the current ecosystem face four challenges from the perspectives of storage, transportation, analysis, and economy.

- **Storage**. The tsunami of genomic data from NGS projects threats public biological databases in terms of space and cost. For example, just after the first 6 months of the 1000 Genomes Project, the raw sequencing data deposited in GenBank's Sequence Read Archive (SRA) division (19) were two times

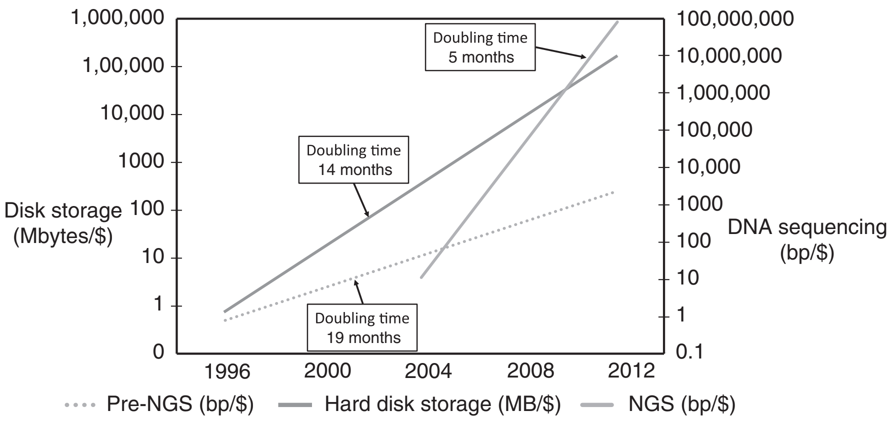

Figure 1.2 Historical trends in storage prices versus DNA sequencing costs (7). Source: Stein et al. 2010. Creative Commons Attribution License 4.0.

larger than all of the data deposited into GenBank in last 30 years (7). Another instance involved NCBI that it announced to discontinue the access service to the high-throughput sequence data due to the unaffordable cost for SRA service (20).

- **Transportation**. The uploading and downloading of huge amounts of data can easily exhaust all the network capacity available to researchers. It is reported that annual worldwide sequencing capacity is currently beyond 13 Pbp (21). Both power users and value-added genome integrators must directly or indirectly download the data from archival databases via the Internet and store copies in local storage systems to analyze them to provide web service. The mirroring of data sets across the network in multiple local storage systems are increasingly cumbersome, error-prone, expensive, and even getting worse when updates are made to databases and all mirrors are needed to be refreshed.
- **Analysis**. The massive amounts of sequence data generated by NGS put the computational burden on traditional analysis significantly. Take sequence assembly of the human genome, for example. Velvet (22), a popular sequential assembly program, needs at least 2 TB memory and several weeks to fully assemble the human genome based on the data from Illumina platform. The single desktop computer is not powerful enough to give us the results in an acceptable time. On the other hand, if we try to cast traditional programs on computing clusters, the coding experience for traditional high-performance computing is not easy to be acquired.
- **Economy**. The load of servers for accessing genome databases and web services usually fluctuates hourly, daily, and seasonally, so large data centers, such as NCBI, UCSC, and other genome data providers, are forced to choose either a cluster to meet average daily requirements or a powerful one to handle peak usage. No matter choosing which option, a large portion of computing resources will stay idle waiting for activities, such as a new large genome data set is submitted, or a major scientific conference is getting close. In addition, as long as the services are online, all the computers require electricity and maintenance, which is not a small amount of the cost.

1.3 BACKGROUND FOR CLOUD COMPUTING AND ITS PROGRAMMING MODELS

A promising solution to address these four challenges mentioned earlier hides in cloud computing, which has been an emerging trend in the scientific community (23). The cloud symbol is often employed to depict the term of “cloud computing” in Internet flowcharts. Based on virtualization technologies, cloud computing provides a variety of services from the hardware level to the application level, and all the services are charged on a pay-per-use basis. Therefore, scientists can have immediate access to needed resources, such as computation power and storage space of large distributed infrastructures, without planning, and release them to save cost as soon as experiments finish.

1.3.1 Overview of Cloud Computing

The general notions in cloud computing can be categorized into two broad types: cloud and cloud technologies. The cloud offers a large pool of easily usable and accessible resources that are scalable to allow optimum utilization (24). A fundamental basis of cloud technologies is virtualization, that is, a single physical machine can host multiple virtual machines (VMs). A VM is a software application that can load a single digital image of resources, often known as a whole system snapshot, and emulate a physical computing environment. In addition, a VM image can be duplicated entirely, including operating system (OS) and its associated applications. Taking the advent of virtualization, the components of infrastructure in the cloud are reusable. At any time point, a particular element in the cloud can be used by a certain user, while at other time points the same element can be employed by other subscribed users. There is no fixed one-to-one relationship between the data or software or physical computing resources. The distinction between traditional computing and virtualization is shown in Figure 1.3. In comparison to the tradition computing, an extra virtualization management layer, Hypervisor, is placed between a physical machine layer and resource images layer. Hypervisor acts as a bridge to translate and transport requests from applications running on VMs to manage physical hardware, such as CPU, memory, hard disks, and network connectivity (25).

Cloud resources for NGS data contain various services, including data storage, data transportation, parallelization of transitional tools, and web services of the data analysis. Basically, cloud services for NGS data can be classified into four categories: Hardware as a Service (HaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Data as a Service (DaaS). More details of these services, including the definitions, cloud-based methods, and NGS applications, will be covered in the next section. With the contributions from open source communities, such as Hadoop, cloud computing becomes more and more popular and practicable in both fields of industry and academy. In brief, users, especially software developers, can pay more attentions on design and arrangement of distributed subtasks for large data sets than for the program deployment on the cloud.

1.3.2 Cloud Service Providers

In this section, three representative cloud service providers will be introduced, that is, Amazon Elastic Compute Cloud, Google App Engine, and Microsoft Azure.

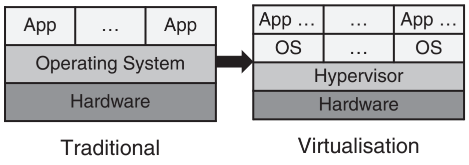

Figure 1.3 Traditional computing versus physical machine with virtualization. Source: O'Driscolla 2013 (25). Reproduced with permission of Elsevier.

Amazon Elastic Compute Cloud (EC2) (26) provides Linux-based or Windows-based virtual computing environments. For users working with NGS data, EC2 offers an integration of public databases embedded in Amazon Web Services (AWS). The integrated databases include the archives from GenBank, Ensembl, 1000 Genomes, Model Organism Encyclopedia of DNA Elements, UniGene, Influenza Virus, and so on. Users can create their Windows-based or Linux-based VMs or load pre-built specific images on the servers they rent. Users just need to upload the VM images to the storage service of EC2, that is, Amazon Simple Storage Service (S3). EC2 only charges users when the allocated VM is alive. As far as we know, the cost for S3 starts at 15 cents per GB per month, and it is 2 cents per hour for using EC2 (27).

Google App Engine (GAE) (28) allows users to build and run web apps on the same systems that are powering Google applications. Various developing and maintenance services are provided by GAE, including fast development and deployment, effortless scalability, and easy administration. Additionally, GAE supports Application Programming Interfaces (APIs) for data management, verification of Google Accounts, image processing, URL fetching, email services, the web-based administration console, and so on. Currently, all applications are allowed to use up to 1 GB storage and other resources free for a month (28).

Microsoft Azure (29) provides users on-demand compute and storage to host, scale, and manage web applications on Microsoft data centers through the Internet. The administrations of applications, such as uploading and updating data, starting and stopping applications, and so on are accessed by a web-based live desktop application. The transfers of every file are protected using Secure Socket Layers combining with users' live ID. Microsoft Azure aims to be a platform to facilitate implementation of SaaS applications.

1.3.3 Programming Models

Cloud programming is about what and how to program on cloud platforms. Although there are many choices of service providers, platforms, and software available in cloud computing, the key point to take advantage of cloud computing technology hinges on the programming model. We take two popular programming models, MapReduce and task programming model, as examples to illustrate some basic principles of how to design cloud applications.

1.3.3.1 MapReduce Programming Model MapReduce is a popular programming model introduced firstly by Google for dealing with data-intensive tasks (30). It allows programmers to apply transformations to each data record and to think in a data-centric manner. In the framework of MapReduce programming model, a job includes three continuous phases: Mapping, Sorting & Merging, and Reducing (31). In mapping, the mapper is the primary unit function, and multiple mappers can be created to fetch and process data records without repeat. The outputs from mapper are all in the form of key/value pair. The sorting & merging phase sorts and groups the outputs according to their keys. In reducing, the reducer is the primary unit function, and multiple reducers can be created to apply further calculations on the grouped outputs. Programs following the MapReduce manner can be automatically executed in parallel on the platforms supporting MapReduce. Developers only need to focus on how to fit their sequential methods into the MapReduce model. The general formation of the mapper is described as follows:

$$\text{map}() :: (\text{key}_1, \text{value}_1) \rightarrow \text{arraylist}\,(\text{key}_2, \text{value}_2) \tag{1.1}$$

The mapper is an initial ingestion and transformation to process input records in parallel. The mapper takes each data record in the form of key/value pair as input and outputs a collection of key/value. The design of key/value can be customized based on users' demand.

The sorting & merging phase sorts the collection of key/value pairs from all mappers in order of the keys. The pairs with the same key are combined and passed to the same reducer.

$$\text{reduce}() :: (\text{key}_2, \text{value}_2) \rightarrow \text{list}\,(\text{value}_3) \tag{1.2}$$

The reducer acts as the aggregation and summarization that all associated records are processed together if necessary as a single entity.

In MapReduce programming model, a complete round of three phases is often considered as a job. Figure 1.4 illustrates the basic workflow of MapReduce programming model. There are several extensions of the basic MapReduce programming model. Three of them are shown in Figure 1.5. As indicated by their names, "map-only" means there is no sorting & merging phase and reducing phases; "map-reduce" is the standard version of MapReduce framework; and "iterative map-reduce" stands for a

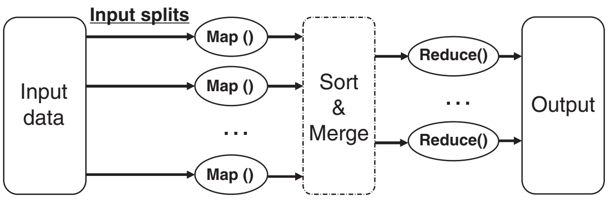

Figure 1.4 The workflow of MapReduce programming model (32). Source: http://hadoop .apache.org. The Apache Software Foundation.

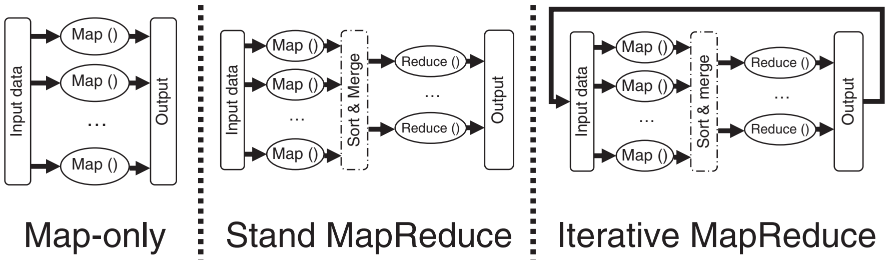

Figure 1.5 Three MapReduce programming models.

multiple rounds of standard map-reduce. An implementation of MapReduce model, Hadoop, will be used as an example to illustrate the MapReduce programming model.

Hadoop Hadoop (32) is an open source software framework for developing data-intensive applications in the cloud. Currently, it can only run on the Linux-based cluster. Hadoop natively supports Java, and it can also be extended to support other languages, such as Python, C, and C++. Hadoop has implemented the model of MapReduce: inputs are partitioned into logical records and processed independently by mappers; results from multiple mappers are sorted and merged into distinct groups; and groups are passed to separate reducers for more calculation. The architecture of Hadoop is shown in Figure 1.6 and the workflow is $Input() \rightarrow Map() \rightarrow Sort()/Merge() \rightarrow Reduce() \rightarrow Output$. The components of Hadoop are organized as a *master-slave* structure. When developing Hadoop applications, we only need to specify three items: Java classes defining key/value pairs, mapper, and reducer (32).

The foundation of Hadoop to support the MapReduce model is the Hadoop Distributed File System (HDFS). HDFS integrates the shared file system as one file system logically. It also provides a Java-based API to handle file operations. The service of HDFS is functionally based on two processes, NameNode and DataNode. NameNode is in charge of control services, and DataNode is in charge of block storage and retrieval services (32). For the scheduling of jobs on each VMs or operating system, Hadoop uses another two processes, TaskTracker and JobTracker. TaskTracker schedules the execution order of each mapper and reducer on slave computing nodes. JobTracker is in charge of job submissions, job monitoring, and distribution of tasks to TaskTrackers (32). In order to obtain high reliability, input data are mirrored into multiple copies in HDFS, referred as “replica.” As long as at least one replica is still alive, TaskTracker is able to continue the job without

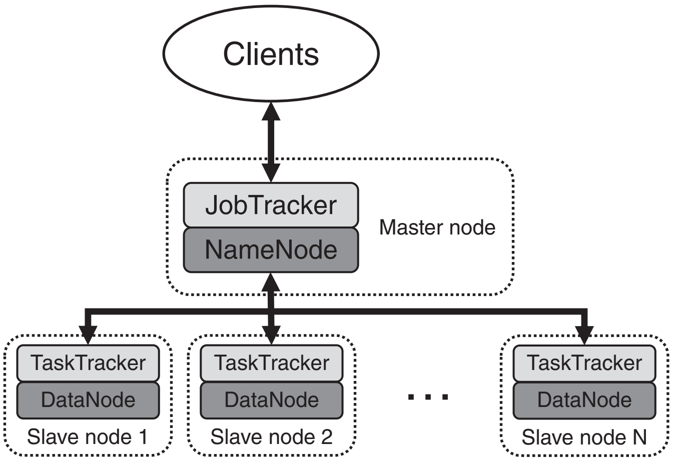

Figure 1.6 The architecture of Hadoop (32). Source: http://hadoop.apache.org. The Apache Software Foundation.

reporting storage failure. Note that master node holds the NameNode and JobTracker services, and slave nodes hold the TaskTracker and DataNode services.

1.3.3.2 Task Programming Model Some scientific problems can be easily split into multiple independent subtasks. Taking BLAST search, for example, a bunch of query sequences can be searched independently if a set of duplicate databases is available and equipped with separate network accessibility. This is the basis for task programming model. The typical framework of task programming model is shown in Figure 1.7.

In the task programming model, subtasks are initially configured by developers and inserted into a task queue. Each entity in this queue contains scheduling information encoded as text messages. A task pool, held by a master node, is used to distribute task entities and coordinate other computing nodes. Task programming model provides a simple way to guarantee fault tolerance: one task can be processed by multiple computing nodes if the task is reported failed, and task pool only deletes the task in the queue when it has been completed. In the following, Microsoft Azure is used as an implementation to illustrate the task programming model.

Microsoft Azure is a cloud service platform provided by Microsoft (29). The architecture of Microsoft Azure is shown in Figure 1.8. It virtualizes hardware resources and abstracts them as Virtual Machines. Any number of conceptually identical VMs

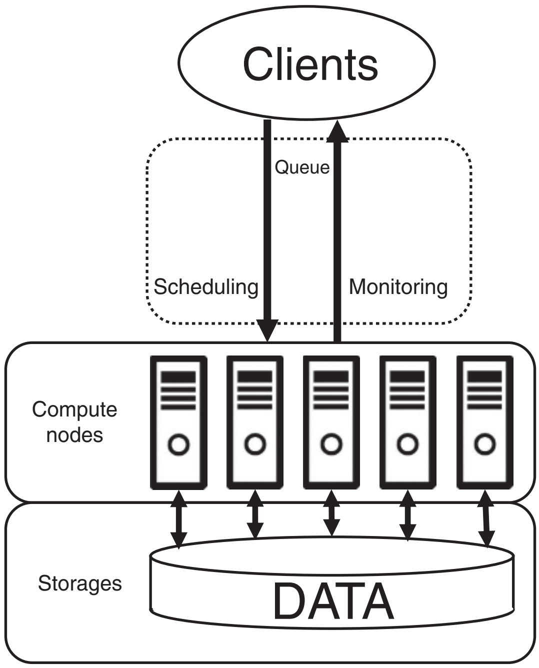

Figure 1.7 The task programming model (33). Source: Gunarathne 2011 (33). Reproduced with permission of John Wiley and Sons.

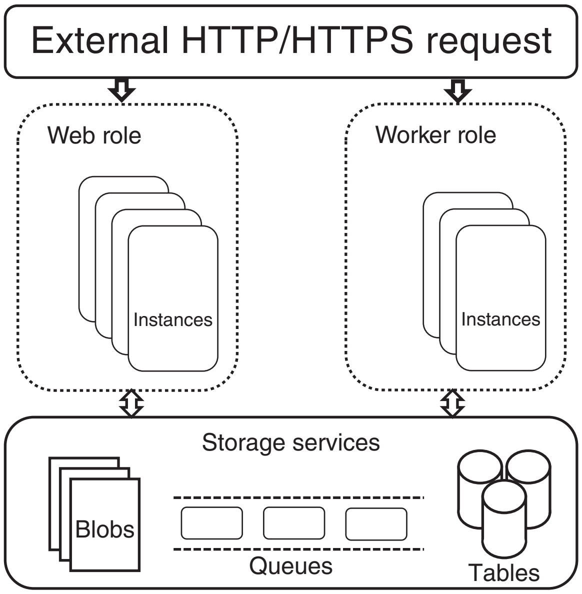

Figure 1.8 The architecture of Microsoft Azure (33). Source: Gunarathne 2011 (33). Reproduced with permission of John Wiley and Sons.

can be readily added to or removed from an application in the abstraction layer above the physical hardware resources, which enhances the administration, availability, and scalability.

A Microsoft Azure application can be divided into several logical components, called "roles", with distinguishing functions. A role contains a particular set of codes, such as a .NET assembly, and an environment where the codes can be executed. Developers can customize the number and scale of instances (VMs) for their applications. There are three types of roles: Web role, Worker role, and VM role. The web role is in charge of front-end web communications, and it is based on Internet Information Services (IIS) compatible technologies, such as ASP.NET, PHP, and Node.js. A worker role, similar to traditional windows desktop environment, performs tasks in the background. The duties include data process and communicate with other role instances. A VM role is used to store the image of Windows Server operating system and can be configured to meet necessary physical environment for running the image. Each role can have multiple VM instances. Unlike Hadoop, instances of worker role can communicate internally or externally in Microsoft Azure.

Most commonly used storage/communication structures in Microsoft Azure are BLOB, Queue, and Azure Table. BLOB, which stands for Binary Large OBject, works as containers, similar to directories in the Unix-based system. Users can set their BLOBs as either public or private and access them by the URLs with account names and access keys. The Queue is a basic structure to support message passing function. The message in the Queue will not disappear permanently until a computing node explicitly deletes it. This feature ensures the fault tolerance as discussed before (34). Azure Table storage is a key/attribute store with a schema-less design where each entity is indexed with row and column, and it can also support query operations, such as traditional database operations.

1.4 CLOUD COMPUTING SERVICES FOR NGS DATA ANALYSIS

In this section, we use case studies to illustrate how the cloud computing services provide support for NGS data analysis. Currently, four main cloud services are available for NGS data, that is, Hardware as a service (HaaS), Platform as a Service (PaaS), Software as a service (SaaS), and Data as a service (DaaS). A summarization of these cloud services is shown in Table 1.1. For SaaS, six methods are used as examples with elaboration on their descriptions, algorithms, and parallel solutions. Four typical biological problems are covered by these six methods, that is, BLAST, comparative genomic, sequence mapping, and SNP detection.

1.4.1 Hardware as a Service (HaaS)

Hardware as a Service, also known as Infrastructure as a service (IaaS), provides users with computing resources, such as storage service and virtualized OS image, through the Internet. Based on the demand from users, HaaS vendors dynamically resize the computing resources and deploy necessary software to build virtual machines. Different users often have different resource requirements, so scalability and customization are two essential features for HaaS. And users only pay for the cloud resources that they use. We briefly introduce several popular HaaS platforms.

AWS is estimated to take 70% of the total HaaS market share. AWS has some offerings, including the Elastic Compute Cloud (EC2) and Simple Storage Service (S3). EC2 provides servers on which users can build VM images. S3 is an online storage service. The HaaS market is changing rapidly with significant fluctuations because HP, Microsoft, Google, and other large companies are all competing for market supremacy. HP releases its cloud platform solution, HPCloud, which integrates servers, storage, networking, and security into an automated system. HPCloud is built on OpenStack, a cloud HaaS software initially developed by Rackspace and NASA. The management of cloud resources in HPCloud is a hybrid service, which combines security and convenience in a private cloud with cost-effectiveness. There are some other HaaS providers, such as Microsoft Azure, Google Compute Engine, and Rackspace. Rackspace is also a hybrid cloud and is able to combine two or more types of cloud, such as private and public, through Virtual Private Networking (VPN) technology typically. The above-mentioned HaaS platforms do share some common features like access to providers’ data center authorized by paying nominal fees and the charge depending on the alive CPU usage, the storage space for the data, and the amount of data transferred.

1.4.2 Platform as a Service (PaaS)

PaaS offers users a platform with necessary software and hardware to develop, test, and deploy cloud applications. In PaaS, VMs can be scaled automatically

TABLE 1.1 Cloud Resources for NGS Data Analysis

Applications	Functions & Descriptions & Availabilities
Hardware as a Service (HaaS):	
Amazon Web Services	Provided by Amazon; http://aws.amazon.com/
HPCloud	Provided by HP; http://www.hpcloud.com/?
Microsoft Azure	Provided by Microsoft; http://www.windowsazure.com/
Google Compute Engine	Provided by Google; http://cloud.google.com/products/compute-engine?
Rackspace	Hybrid Cloud; http://www.rackspace.com/
Platform as a Service (PaaS):	
Google App Engine	http://developers.google.com/appengine/?
Microsoft Azure	http://www.windowsazure.com/?
MapReduce/Hadoop	http://hadoop.apache.org/?
Software as a Service (SaaS):	
AzureBlast (35)	BLAST (a parallel BLAST running on the cloud computing platform of Microsoft Azure)
CloudBLAST (36)	BLAST (a cloud-based implementation of BLAST) http://ammatsun.acis.ufl.edu/amwiki/index.php/CloudBLAST_Project
BlastReduce (37)	BLAST (Hadoop-based BLAST) http://www.cbcb.umd.edu/software/blastreduce/
RSD (38)	Comparative genomics (reciprocal smallest distance algorithm for ortholog detection on EC2) http://roundup.hms.harvard.edu
CloudBurst (39)	Genomic sequence mapping (highly sensitive short read mapping with MapReduce) http://cloudburst-bio.sourceforge.net
SeqMapReduce (40)	Genomic sequence mapping (parallelizing sequence mapping by using Hadoop) http://www.seqmapreduce.org
CloudAligner (41)	Genomic sequence mapping (fast and full-featured MapReduce-based tool for sequence mapping) http://cloudaligner.sourceforge.net
SEAL (42)	Genomic sequence mapping (short read pair mapping and duplicate removal tool by using Hadoop) http://biodoop-seal.sourceforge.net/.
Crossbow (43)	Genomic sequence analysis (read mapping and SNP calling using cloud computing)
CrossTSS (44)	Genomic sequence analysis (a parallel haplotype block partition and SNPs selection method by using the Hadoop)
Contrail (45)	Genomic sequence analysis (cloud-based *de novo* assembly of large genomes) http://contrail-bio.sourceforge.net
CloudBrush (46)	Genomic sequence analysis (a genome assembler based on string graphs and MapReduce) https://github.com/ice91/CloudBrush
Myrna (47)	RNA sequencing analysis (differential gene expression tool for RNA-Seq) http://bowtie-bio.sourceforge.net/myrna

TABLE 1.1 (*Continued*)

Applications	Functions & Descriptions & Availabilities
Eoulsan (48)	RNA sequencing analysis (a modular and scalable framework based on the Hadoop) http://transcriptome.ens.fr/eoulsan/
FX (49)	RNA sequencing analysis(RNA-Seq analysis tool) http://fx.gmi.ac.kr
HadoopBAM (50)	Sequence file management (an integration layer between analysis applications and BAMfiles) http://sourceforge.net/projects/hadoop-bam
SeqWare (51)	Sequence file management (query engine supporting databasing information for large genomes) http://seqware.sourceforge.net
GATK (52)	Sequence file management (a gene analysis toolkit for next-generation resequencing data) http://www.broadinstitute.org/gatk/
Data as a Service (DaaS):	
AWS Public Data Sets	(Cloud-based archives of GenBank, 1000 Genomes, and so on) http://aws.amazon.com/publicdatasets

and dynamically to meet applications' demands. Because the deployment and assignment of hardware are in a transparent manner, users can pay more attentions on the development of cloud-based programs. Typically, the environment delivered by PaaS comes with programming language execution environments, web servers, and databases. Some popular platforms have been introduced in the beginning, such as Google App Engine, Microsoft Azure, and MapReduce/Hadoop. When considering cloud-based databases, DaaS can be also treated as an instance of PaaS. Here, we separate DaaS from PaaS and will discuss DaaS later.

1.4.3 Software as a Service (SaaS)

SaaS provides on-demand software as web services and facilitates remote access to data analyses in various types. The analysis of NGS data involves many biological issues, such as sequence mapping, sequence alignment, sequence assembly, expression analysis, sequence analysis, orthology detection, functional annotation of personal genomes, detection of epistatic interactions, and so on (25). SaaS eliminates the necessity of complicated local deployment, simplifies software maintenances, and ensures up-to-date cloud-based services for all possible users with access to the Internet. Since it is impractical to cover all cloud-based NGS data analysis tools available nowadays, four representative categories are carefully selected to elaborate the thinking in the cloud for solving problems that arise from NGS data.

1.4.3.1 BLAST Basic Local Alignment Search Tool (BLAST) (35) is one of the most widely used sequence analysis programs provided by NCBI. Meaningful information from the query sequence can be extracted by comparing it to the NCBI databases using BLAST. The pairwise comparison is trivial when only limited number of sequences is needed to be compared, but the number of sequences in NCBI's databases are extremely large. For instance, 361 billion nucleotide bases were reported in Reference Sequence (RefSeq) Database up to November 10, 2013. Without doubt, it is computational intensive even when one query is submitted to a huge database by using pairwise comparison. Several cloud-based applications have been proposed to parallel BLAST on commercial cloud platforms. The basic strategies of them are very similar. Because the queries of sequences are independent, they can be executed simultaneously on a set of separate computers with a partial or complete database. In the following, two cloud applications, AzureBlast (53) and CloudBLAST (36), are discussed to illustrate the idea.

AzureBlast Lu et al. (53) proposed a parallel BLAST, named AzureBlast, running on the Microsoft Azure. The workflow of AzureBlast is shown in Figure 1.9. Instead of partitioning the database into segmentations, they use a query-segmentation data-parallel pattern to split the query sequences into several disjoint sets. The reason for this is that the queries on segmentations need less communication among instances than the query on several parts of the database. Given some sequences as the input, AzureBlast partitions the input sequences into multiple files and allocates them to worker instances to start the comparisons. The results are merged from all worker instances. The experiments of AzureBlast demonstrate that Microsoft Azure can very well support the BLAST based on its scalable and fault-tolerant computation and storage services (53).

CloudBLAST Matsunaga et al. (36) proposed a WAN-based implementation of BLAST, called CloudBLAST. In CloudBLAST, the parallelization, deployment, and management of applications are built and evaluated on Hadoop platform. Similar to

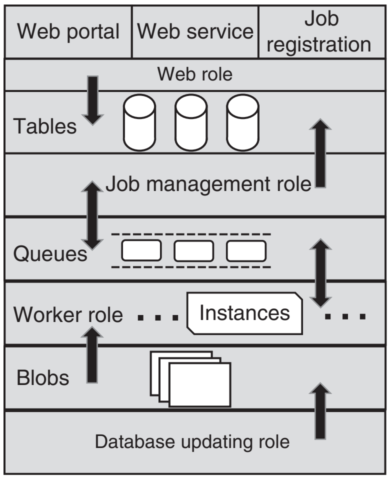

Figure 1.9 The workflow of AzureBlast (53).

AzureBlast, input query sequences are split at first and then the grouped sequences are passed to mappers to run BLAST program separately. The results from mappers are stored on a local disk and combined as the final results. Demonstrated by the experiments of CloudBLAST, cloud-based applications built on Internet-connected resources for bioinformatics issues can be considerably efficient. CloudBLAST's performance was experimentally contrasted against a publicly available tool, mpiBLAST (54), on the same cloud configuration. mpiBLAST is a free parallel implementation of NCBI BLAST running on clusters with job-scheduling software such as PBS (Portable Batch System). By using 64 processors, both tools gained nearly equivalent performance with speedups (31) of 57 of CloudBLAST against 52.4 of mpiBLAST (36), respectively.

1.4.3.2 Comparative Genomics Comparative genomics is a study of understanding functional similarities and differences as well as evolutionary relationships between genomes by comparing genomic features, such as DNA sequence, genes, and regulatory sequences, across different biological species or strains. One computationally intensive application in comparative genomics is the Reciprocal Smallest Distance algorithm (RSD) (38). RSD is used to detect orthologous sequences between multiple pairs of genomes. It has three steps: (i) employ BLAST to generate a set of hits between query sequences and references, (ii) use alignment tools on each protein sequence and take PAML (38) to obtain the maximum likelihood estimation of the number of amino acid substitutions, and (iii) call BLAST again to re-calculate the maximum likelihood distance to determine whether the pair of sequences is correct orthologous pair or not. Wall et al. (38) proposed a cloud-based tool, named RSD-cloud, by fitting legacy RSD into MapReduce model on EC2. RSD-cloud has two primary phases, that is, BLAST and estimation of evolutionary distance. In the first phase, mappers use BLAST to generate hits for all genomes. In the second phase, mappers conduct ortholog computation to estimate orthologs and evolutionary distances for all genomes. As shown in Figure 1.10, two blocks in step 2 illustrate the above two paralleled phases. All results from RSD-cloud directly go into Amazon S3. Experiments showed that it is able to run more than 300,000 RSD-cloud processes within the EC2 to compute the orthologs for all pairs of 55 genomes by using 100 high-capacity computing nodes (38). The total computation time was less than 70 hours and the cost was $6,302 USD.

1.4.3.3 Genomic Sequence Mapping Genomic sequence mapping aims to locate the relative positions of genes or DNA fragments on the reference chromosomes. Generally, there are two types of genomic sequence mapping: (i) genetic mapping, using classical genetic techniques, such as pedigree analysis, to depict the features of genome, and (ii) physical mapping, using modern molecular biology techniques for the same goal. Current cloud-based solutions for genomic sequence mapping belong to the second type. Similar to BLAST, genomic sequence mapping can also be paralleled in terms of independence of the sequence queries although extra processes may be needed. In the following, two cloud tools, CloudBurst (39) and CloudAligner (41), are used to illustrate the cloud-based solutions for genomic sequence mapping.

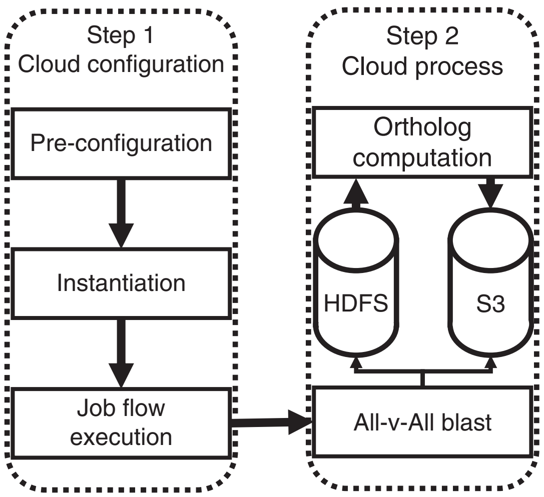

Figure 1.10 Workflow of RSD using the MapReduce framework on the EC2 (38). Source: Wall, http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-259. Used under CC BY 2.0 http://creativecommons.org/licenses/by/2.0/.

CloudBurst Schatz (39) designed a parallel algorithm, named CloudBurst, which is a seed-and-extend read mapping algorithm on Hadoop platform based on a popular read mapping program, RMAP (55). According to MapReduce model, CloudBurst modifies RMAP to run on multiple machines in parallel. The workflow of CloudBurst with two phases, Map phase and Reduce phase, is shown in Figure 1.11. The key/value pairs generated by mappers have the following format, <reads' and references' indexes, *k*-mers of reads and references>. Reducers execute end-to-end alignments between reads and reference sequences sharing the same *k*-mers. Final results are converted into text files with the standard format as RMAP did, so CloudBurst can replace RMAP in other pipelines. CloudBurst's running time scales near linearly as the number of processors increases. In a configuration with 24-processor cores, CloudBurst achieved up to 30 times faster than RMAP executed on a single core given an identical set of alignments as input (39).

CloudAligner Nguyen et al. (41) developed CloudAligner, a MapReduce-based application to handle long read mapping. It only uses map phase to locate the reads on references. The workflow is shown in Figure 1.12. CloudAligner accepts two types of input files, that is, read files and reference files. CloudAligner first splits read files into a bunch of chunks with limited size and distributes them to multiple mappers. Mappers then align the reads to the references. CloudAligner is able to deal with multiple mapping types, including mismatch mapping, bisulfite mapping, and pair-end mapping, and it is also compatible with various types of input, such as fastq and SAM. The experimental results demonstrated that CloudAligner can achieve significant improvement in terms of efficiency by using the MapReduce model without reduce phase.

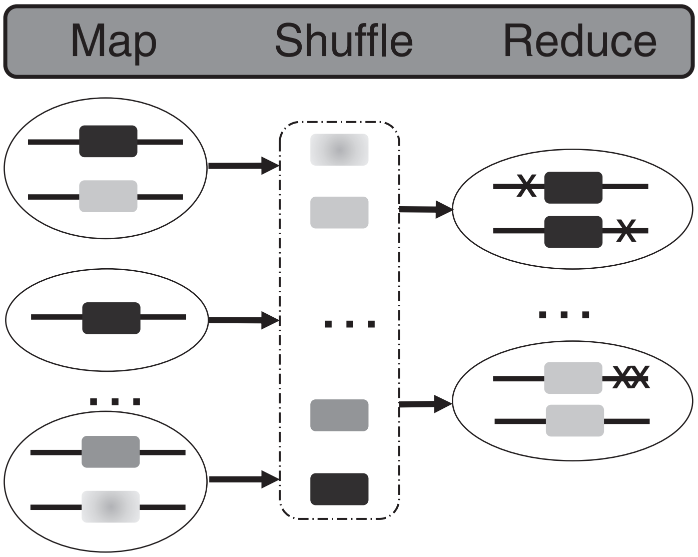

Figure 1.11 The overview of CloudBurst (39). Source: Schatz 2009 (39). Reproduced with permission of Oxford University Press.

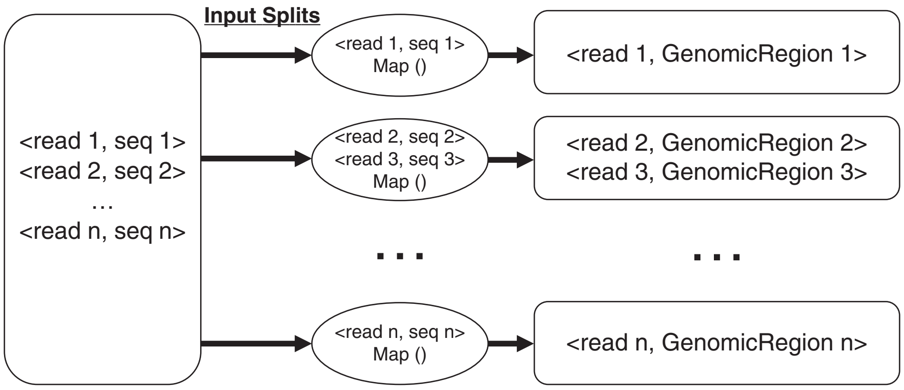
Input Splits
<read 1, seq 1>
<read 2, seq 2>
…
<read n, seq n>
<read 1, seq 1> Map ()
<read 2, seq 2> <read 3, seq 3> Map ()
<read n, seq n> Map ()
<read 1, GenomicRegion 1>
<read 2, GenomicRegion 2>
<read 3, GenomicRegion 3>
<read n, GenomicRegion n>

Figure 1.12 The workflow of CloudAligner (41). Source: Nguyen, http://bmcresnotes .biomedcentral.com/articles/10.1186/1756-0500-4-171. Used under CC BY 2.0 http:// creativecommons.org/licenses/by/2.0/.

1.4.3.4 SNP Detection SNP detection is used to scan for single-nucleotide polymorphism (SNP), which is a DNA sequence variation occurring commonly within a population. A single nucleotide in the genome is defined as SNP when it differs between members of a biological species or paired chromosomes. These genetic variations are considered to be associated with many genetic diseases (56, 57). A central challenge of SNP detection based on NGS data is the sequence alignment that massive sequence reads must be compared to the references. Similar to the previous comparison problems, SNP detections on multiple sequence areas are independent of each other. In the following, an application, Crossbow, is used to show the cloud solution for SNP detection.

Crossbow Langmead et al. (43) proposed a Hadoop-based tool, Crossbow, which uses alignment to detect SNP for the whole human genomes. Crossbow combines two methods together, that is, a short read aligner, Bowtie (58), and an SNP caller, SOAPsnp (59). Crossbow has three phases: Map phase, Sorting & merging phase, and Reduce phase. In Map phase, mappers align short reads to the reference sequences by using Bowtie to produce a stream of alignment pairs. There are two keys for each pair: primary key and secondary key. The primary key is the identifier of the chromosome, and the secondary key is the beginning location of the queried sequence on the chromosome. The value of each pair is the aligned sequence and quality scores. In Sorting & merging phase, Hadoop platform automatically groups these pairs according to the primary key and sorts them in a particular order based on the secondary key. In Reduce phase, reducers simply call SOAPsnp to perform SNP detection. As shown in the experiments, Crossbow consumed only 3 hours to finish the alignments and SNP detection for a Han Chinese male genome with 38-fold coverage by using a compute cluster with 320 cores.

1.4.4 Data as a Service (DaaS)

DaaS aims to deliver data from multiple sources in various formats as a service, which can be accessed through the Internet. Bioinformatic tools heavily depend on the data, especially those analyses about the genome, which are fundamentally crucial for downstream analyses. DaaS enables dynamic data access on demand, so users can remotely obtain the data via Internet as if they are using the data in a local file system. In addition, the sharing of data accelerates the progress of large projects for the real-time collaboration with other researchers. An example of DaaS is the AWS, which provides a centralized repository of public data, including archives of GenBank, Ensembl, 1000 Genomes, Model Organism Encyclopedia of DNA Elements, UniGene, Influenza Virus, and so on. In fact, AWS also contains many public data sets for other scientific fields, such as astronomy, chemistry, climate, and economics. All public data sets in AWS are delivered as services, and thus they can be seamlessly integrated into cloud-based applications (27).

1.5 CONCLUSIONS AND FUTURE DIRECTIONS

In this chapter, we first discussed some challenges proposed by NGS technologies in terms of storage, transportation, analysis, and economy. All these challenges are caused by the tsunami of genomic data generated by NGS platforms. It is becoming impossible for a single desktop to run sequential approaches to obtain results in an acceptable time. Cloud computing is a promising solution to rescue researchers for computationally intensive problems. Two categories of cloud programming models, MapReduce and task programming models, were elaborated to illustrate the parallel idea of current cloud platforms. A summary of cloud resources for NGS data analysis was listed in terms of four cloud services, that is, HaaS, PaaS, SaaS, and DaaS. Six cloud-based methods of SaaS for four NGS data analysis problems were revised, which were used as an inspiration for how to perform transformation of sequential programs onto cloud platforms. However, not all data-intensive algorithms can be easily transferred to cloud environments. Therefore, suitable parallel strategies, more powerful programming models, frameworks, and cloud platforms are urgently needed. There are three basic parallel tactics commonly used in developing cloud-based applications:

1. Straightforward Parallel. Many bioinformatics tasks, such as BLAST, are explicitly data independent. So they can run on cloud platforms with only minor modifications. The speedup of this parallel type is near linear with respect to the computation power of a computing cluster.
2. Iterative Parallel. Multiple MapReduce rounds or executions of a program are another solutions for some complicated jobs, like RSD. Additional operations about how to store, allocate, and collect intermediate temporary data may also be carefully considered. How to design an efficient method is a fundamental challenge, since temporary data could be too huge to be processed during the execution.
3. Data Transformation Parallel. For some sophisticated problems, the conversion of data and algorithm itself are needed to take the advantage of cloud computing. An example is the one in Reference 60. Although we did not introduce it in this chapter, it is still worthy to know that we can change the format of inputs wisely to facilitate the development of cloud-based solutions. The challenge lying in parallel tactic is how to design a suitable data formation without any or with few compromises of accuracy or efficiency of legacy programs.

Concerning bioinformatics, not limited in the analysis of NGS data, cloud computing is still at its early stage. More powerful and readily usable cloud platforms and programming models are aspired to be invented and invested to conquer complex scientific issues. Ultimately, all scientists will benefit by taking advantages of this new computation power in all areas.

REFERENCES

1. Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors. Proc Natl Acad Sci U S A 1977;74(12):5463–5467.
2. Quail MA, Smith M, Coupland P, Otto TD, Harris SR, Connor TR, Bertoni A, Swerdlow HP, Gu Y. A tale of three next generation sequencing platforms: comparison of ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics 2012;13(1):341.
3. Shanker A. Genome research in the cloud. OMICS 2012;16(7-8):422–428.
4. Benson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW. GenBank. Nucleic Acids Res 2012;41(D1):D36–D42. DOI: 10.1093/nar/gks1195.

5. Brooksbank C, Camon E, Harris MA, Magrane M, Martin MJ, Mulder N, O'Donovan C, Parkinson H, Tuli MA, Apweiler R et al. The European Bioinformatics Institute's data resources. Nucleic Acids Res 2003;31(1):43–50.
6. Shumway M, Cochrane G, Sugawara H. Archiving next generation sequencing data. Nucleic Acids Res 2010;38 Suppl 1:870–871.
7. Stein LD et al. The case for cloud computing in genome informatics. Genome Biol 2010;11(5):207.
8. Moore GE et al. *Cramming More Components Onto Integrated Circuits*. New York; McGraw-Hill; 1965.
9. Walter C. Kryder's law. Sci Am 2005;293(2):32–33.
10. Reynolds C. As we may communicate. ACM SIGCHI Bull 1998;30(3):40–44.
11. Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, Berka J, Braverman MS, Chen Y-J, Chen Z et al. Genome sequencing in microfabricated high-density picolitre reactors. Nature 2005;437(7057):376–380.
12. Bennett S. Solexa Ltd. Pharmacogenomics 2004;5(4):433–438.
13. McKernan KJ, Peckham HE, Costa GL, McLaughlin SF, Fu Y, Tsung EF, Clouser CR, Duncan C, Ichikawa JK, Lee CC et al. Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding. Genome Res 2009;19(9):1527–1541.
14. Internet Archive. Available at http://www.archive.org/. Accessed 2016 Mar 19.
15. Illumina. Available at http://www.illumina.com/. Accessed 2016 Mar 19.
16. Pacific Biosciences. Available at http://www.pacificbiosciences.com/. Accessed 2016 Mar 19.
17. Helicos Biosciences Corporation. Available at http://www.helicosbio.com/. Accessed 2016 Mar 19.
18. Ion Torrent. Available at http://www.iontorrent.com/. Accessed 2016 Mar 19.
19. The 1000 Genomes Project. Available at http://www.1000genomes.org/. Accessed 2016 Mar 19.
20. NCBI News. Available at http://www.ncbi.nlm.nih.gov/About/news/16feb2011. Accessed 2016 Mar 19.
21. Mediawiki. Available at http://sour-ceforge.net/apps/mediawiki/jnomics. Accessed 2016 Mar 19.
22. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 2008;18(5):821–829.
23. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 2009;25(6):599–616.
24. Vaquero LM, Rodero-Merino L, Caceres J, Lindner M. A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev 2008;39(1):50–55.
25. O'Driscoll A, Daugelaite J, Sleator RD. 'Big data', Hadoop and cloud computing in genomics. J Biomed Inform 2013;46(5):774–781.
26. Amazon Elastic Compute Cloud (EC2). Available at https://aws.amazon.com/ec2/.
27. Fusaro VA, Patil P, Gafni E, Wall DP, Tonellato PJ. Biomedical cloud computing with Amazon Web Services. PLoS Comput Biol 2011;7(8):1002147.
28. Google App Engine. Available at http://appengine.google.com. Accessed 2016 Mar 19.
29. Windows Azure. Available at http://www.windowsazure.com/. Accessed 2016 Mar 19.

30. Jin C, Buyya R. MapReduce programming model for. NET-based cloud computing. In: Sips H, Epema D, Lin H-X, editors. *Euro-Par 2009 Parallel Processing*. Volume 5704 of Lecture Notes in Computer Science. Berlin Heidelberg: Springer-Veralg; 2009. p 417–428.
31. Guo X, Ding X, Meng Y, Pan Y. Cloud computing for de novo metagenomic sequence assembly. In: Cai Z, Eulenstein O, Janies D, Schwartz D, editors. *Bioinformatics Research and Applications*. Volume 7875 Lecture Notes in Computer Science. Berlin Heidelberg: Springer-Veralg; 2013. p 185–198.
32. Hadoop. Available at http://hadoop.apache.org. Accessed 2016 Mar 19.
33. Gunarathne T, Wu T-L, Choi JY, Bae S-H, Qiu J. Cloud computing paradigms for pleasingly parallel biomedical applications. Concurr Comput Pract Exp 2011;23(17): 2338–2354.
34. Redkar T, Guidici T, Meister T. *Windows Azure Platform*. Volume 1. Berkeley (CA): Apress; 2011.
35. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol 1990;215(3):403–410.
36. Matsunaga A, Tsugawa M, Fortes J. CloudBlast: combining MapReduce and virtualization on distributed resources for bioinformatics applications. *eScience, 2008. eScience'08. IEEE 4th International Conference On*; 2008. p 222–229.
37. Schatz MC. *BlastReduce: High Performance Short Read Mapping with MapReduce*. University of Maryland; 2008. Available at http://cgis.cs.umd.edu/Grad/scholarlypapers/papers/MichaelSchatz.pdf. Accessed 2016 Mar 19.
38. Wall DP, Kudtarkar P, Fusaro VA, Pivovarov R, Patil P, Tonellato PJ. Cloud computing for comparative genomics. BMC Bioinformatics 2010;11(1):259.
39. Schatz MC. CloudBurst: highly sensitive read mapping with MapReduce. Bioinformatics 2009;25(11):1363–1369.
40. Li Y, Zhong S. SeqMapReduce: software and web service for accelerating sequence mapping. Critical Assessment of Massive Data Anaysis (CAMDA) 2009; 2009.
41. Nguyen T, Shi W, Ruden D. CloudAligner: a fast and full-featured MapReduce based tool for sequence mapping. BMC Res Notes 2011;4(1):171.
42. Pireddu L, Leo S, Zanetti G. Seal: a distributed short read mapping and duplicate removal tool. Bioinformatics 2011;27(15):2159–2160.
43. Langmead B, Schatz MC, Lin J, Pop M, Salzberg SL. Searching for SNPs with cloud computing. Genome Biol 2009;10(11):134.
44. Hung C-L, Lin Y-L, Hua G-J, Hu Y-C. CloudTSS: A TagSNP selection approach on cloud computing. In: Kim TH et al., editors. *Grid and Distributed Computing*. Volume 261. Berlin Heidelberg: Springer-Verlag; 2011. p 525–534.
45. Schatz M, Sommer D, Kelley D, Pop M. Contrail: assembly of large genomes using cloud computing. CSHL Biology of Genomes Conference; 2010.
46. Chang Y-J, Chen C-C, Chen C-L, Ho J-M. A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework. BMC Genomics 2012;13 Suppl 7:28.
47. Langmead B, Hansen KD, Leek JT et al. Cloud-scale RNA-sequencing differential expression analysis with Myrna. Genome Biol 2010;11(8):83.
48. Jourdren L, Bernard M, Dillies M-A, Le Crom S. Eoulsan: a cloud computing-based framework facilitating high throughput sequencing analyses. Bioinformatics 2012;28(11):1542–1543.

49. Hong D, Rhie A, Park S-S, Lee J, Ju YS, Kim S, Yu S-B, Bleazard T, Park H-S, Rhee H et al. FX: an RNA-Seq analysis tool on the cloud. Bioinformatics 2012;28(5):721–723.
50. Niemenmaa M, Kallio A, Schumacher A, Klemelä P, Korpelainen E, Heljanko K. Hadoop-BAM: directly manipulating next generation sequencing data in the cloud. Bioinformatics 2012;28(6):876–877.
51. O'Connor BD, Merriman B, Nelson SF. SeqWare Query Engine: storing and searching sequence data in the cloud. BMC Bioinformatics 2010;11 Suppl 12:2.
52. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M et al. The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 2010;20(9):1297–1303.
53. Lu W, Jackson J, Barga R. AzureBlast: a case study of developing science applications on the cloud. Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM; 2010. p 413–420.
54. Darling A, Carey L, Feng W-C. The design, implementation, and evaluation of mpi-BLAST. Proceedings of ClusterWorld 2003; 2003.
55. Smith AD, Xuan Z, Zhang MQ. Using quality scores and longer reads improves accuracy of Solexa read mapping. BMC Bioinformatics 2008;9(1):128.
56. Patil N, Berno AJ, Hinds DA, Barrett WA, Doshi JM, Hacker CR, Kautzer CR, Lee DH, Marjoribanks C, McDonough DP et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. Science 2001;294(5547):1719–1723.
57. Guo X, Meng Y, Yu N, Pan Y. Cloud computing for detecting high-order genome-wide epistatic interaction via dynamic clustering. BMC Bioinformatics 2014;15(1):102.
58. Langmead B, Trapnell C, Pop M, Salzberg SL et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 2009;10(3):25.
59. Li R, Li Y, Fang X, Yang H, Wang J, Kristiansen K, Wang J. SNP detection for massively parallel whole-genome resequencing. Genome Res 2009;19(6):1124–1132.
60. Estrada T, Zhang B, Cicotti P, Armen R, Taufer M. A scalable and accurate method for classifying protein–ligand binding geometries using a MapReduce approach. Comput Biol Med 2012;42(7):758–771.