genes

MDPI

Perspective

Computational Strategies for Scalable Genomics Analysis

Lizhen Shi [1] **and Zhong Wang** [2,3,4,*]

[1] Department of Computer Science, Florida State University, Tallahassee, FL 32304, USA; lizhen9.shi@gmail.com
[2] US Department of Energy, Joint Genome Institute, Walnut Creek, CA 94598, USA
[3] Environmental Genomics and Systems Biology Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[4] School of Natural Sciences, University of California at Merced, Merced, CA 95343, USA
* Correspondence: zhongwang@lbl.gov

Received: 10 October 2019; Accepted: 3 December 2019; Published: 6 December 2019

Abstract: The revolution in next-generation DNA sequencing technologies is leading to explosive data growth in genomics, posing a significant challenge to the computing infrastructure and software algorithms for genomics analysis. Various big data technologies have been explored to scale up/out current bioinformatics solutions to mine the big genomics data. In this review, we survey some of these exciting developments in the applications of parallel distributed computing and special hardware to genomics. We comment on the pros and cons of each strategy in the context of ease of development, robustness, scalability, and efficiency. Although this review is written for an audience from the genomics and bioinformatics fields, it may also be informative for the audience of computer science with interests in genomics applications.

Keywords: scalable genomics analysis; big data; high performance computing; cloud computing

1. Introduction

The successful completion of the human genome project led to a revolution in DNA sequencing. In the ten years following the completion of the first human genome in 2003, several sequencing technologies, collectively called Next-Generation Sequencing (NGS), underwent several rounds of development and displacement. Each round brought capacity increases of 100–1000 fold (reviewed in [1]). By 2019, two types of sequencing technology remain popular: Short-read and long-read. If we think of a genome as a book, short-read sequencers read this book sentence-by-sentence with few errors, while long-read sequencers read it paragraph-by-paragraph, typically with a higher error rate. Illumina dominates the short-read sequencing market and two companies presently compete in the long read sequencer market, Pacific Biosciences and Oxford Nanopore. All platforms are massively parallel sequencers. Currently, the majority of sequencing data are from short-read technologies because they have much higher throughput and lower cost per base. Thanks to these powerful sequencing technologies, genomics, like many other disciplines, has entered the big data era [2]. In FY2018, the Department of Energy Joint Genome Institute (DOE JGI) generated 200 Terabases (Tb) of sequence data, enough to sequence one human genome over 3000 times at 20× coverage. The total amount of sequence of the first release of the UK BioBank project (50,000 participants, or about 1/10 of the total) is already over 50 TB [3].

Unlike other domains, where big data are characterized by the four "V"s: Volume (for large scale), variety (for different forms), velocity (for fast streaming) and veracity (for data uncertainty), genomics big data also has a "U" characteristic, meaning the majority of genomics data are unstructured. In

addition, data volume, variety, and veracity often increase as genomics data flows through various data analytic steps. It is not uncommon to need disk space 100–200 times the input for storing temporary data. Software tools, having very diverse input requirements and functionalities, are needed to extract and reduce the information contained in the sequence data, and these tools inevitably introduce noises and biases.

The exponential growth of genomics data poses a great challenge for data management and analysis. We will not discuss data storage and management problems here. Interested readers are encouraged to consult the excellent review [4]. In this review, we will discuss general strategies of scalable genomics solutions without focusing on algorithms for specific genomics applications. We will discuss these solutions in the context of four criteria: Ease of development, robustness, scalability, and efficiency. Here, scalability refers to the capability of a bioinformatic tool to handle bigger data/problems when additional resources are added. Efficiency indicates how well a tool utilizes the computational resources of the system. We use scalability in combination with efficiency to measure how well a bioinformatic tool effectively uses increasing numbers of parallel processing elements (nodes, CPUs, cores, processes, threads, etc.) for processing a growing amount of data. It is important to note that, besides parallelization that we are going to discuss below, there are many generic factors that could impact the performance of genomic solutions, including compilers, programming models, software-hardware optimization, etc.

2. Scalable Strategies

2.1. Shared-Memory Multicore Architecture

Given that many genomics research groups do not have seasoned software engineers for distributed software development, exploiting single servers with a large amount of RAM and CPU cores is a straightforward solution for large-scale sequence analysis. My group initially invested in big memory nodes with Terabytes of RAM. The cache-coherent non-uniform memory access architecture of the XSEDE resource Blacklight, housed at the Pittsburgh Supercomputing Center (PSC), with up to 16 TB of RAM, enables very large plant genome assemblies such as the hexaploid wheat genome [5]. Similarly, an SGI UV200 machine with 7 TB of shared RAM also enables wheat genome assembly (38 days on 64 CPUs) [6]. Amazon Web Services (AWS) also offers large memory EC2 instances with up to 4 TB and 128 cores (X1e instances) and much larger memory footprints are in alpha testing.

In these shared memory supercomputers with large memory and lots of cores, parallelism is usually achieved by multi-threading, i.e., dividing a process into multiple threads and executing them in parallel. As memory and other resources are shared among threads, threads can read and write to the shared memory without the need for special mechanisms of communication. Pthread [7] and OpenMP [8] are two implementations of multi-threading at different levels. Pthreads, the POSIX standard thread library, is a low-level application programming interface (API) for working with threads, whereas OpenMP gives programmers higher level threading options that greatly simplify programming and debugging. Consequently, OpenMP is the predominant threading model on shared memory systems for scientific computing. Accordingly, there are more OpenMP-based tools (SPAdes [9–11]) than Pthread-based tools (ALLPATHS-LG [12], SOAPdenovo [13]) on genomic analysis.

These large memory systems enable rapid results without extra software development time. However, the cost of upgrading a node increases exponentially with memory size. Furthermore, there are physical limits to how much memory or cores can be added to a node.

2.2. Special Hardware

The increasing demand for power-efficient, high-performance computing spurred a revolution in computer architectures over the last couple of decades, as specialized processing units emerged to improve the efficiency of parallel tasks. In this environment, the classical host processor delegates the

execution of the computationally intensive parts of the job to co-processors, such as field-programmable gate arrays (FPGA) [14], graphics-processing units (GPU) [15], and tensor processing unit (TPU) [16], to speed up the overall execution. These special hardware architectures are now being applied to genomics data analysis with remarkable success. For example, Falcon Computing developed an FPGA-based solution [17] that speeds up the genome analysis tool kit (GATK) [18], a genomics variant analysis suite, by 50 times. GPU has a long history in computational biology for special applications such as molecular dynamics simulation [19,20], and recently sees an increasing application in NGS data analysis (reviewed in [21]). As the genomics field is rapidly adopting deep learning technologies (reviewed in [22]), GPUs and TPUs are expected to speed up more genomics applications such as AlphaFold [23].

These special hardware architectures greatly increased the amount of parallelism that a machine can exploit. However, there are several limitations including availability, difficulty scaling on heterogeneous systems, and the need to port existing CPU-based algorithms to these systems. In addition, training large deep neural networks on GPUs/TPCs could be very cost-prohibitive [24].

2.3. Multi-Node HPC Scalability

For on-premise hardware scaling, one could either scale up by upgrading the existing nodes with more capacity (cores, memory, co-processing unit, storage, etc.) as we discussed above, or scale out—by adding more nodes to form a high-performance computing cluster (HPC).

Message Passing Interface (MPI) [25], the *de facto* industry standard for distributed memory systems, is a language-independent communications protocol that uses a message-passing paradigm to share the data and state among a set of cooperative processes running on an HPC cluster. As it takes the advantage of data locality, MPI-based implementation yields great computing performance compared with alternatives [26]. MPI-based NGS sequence analysis tools, including read aligner such as pBWA [27] and assembler such as Ray [28], can scale up to hundreds of thousands cores on a HPC cluster.

Partitioned Global Address Space (PGAS) [29] is a distributed shared-memory programming model that combines the advantages of the shared-memory programming paradigm and the performance of the message passing programming paradigm. Unified Parallel C (UPC) [30] and UPC++ [31] are C and C++ extensions of PGAS model, repsectively, by combining the advantages of the PGAS and C/C++ language features such as templates, object-oriented design, operator overloading, and lambda functions. These advantages have brought noticeable performance gains in a few challenging genomic problems including metagenome assembly: UPC-based tools like Meta-HipMer [32] can assemble a 2.6 TB metagenome dataset in just 3.5 h with 512 nodes.

The biggest drawback of MPI and the PGAS languages is their programmability, as they require experienced software engineers to take care of fine-grained control over mechanisms such as memory locality, data communication, and tasks synchronization. These inevitably drive up the development and maintenance costs. Another potential drawback of large-scale runs is fault-tolerance; failure of one process could lead to the failure of the entire application. Some recent community efforts aim to combine the ease of programming such as Python with the superior efficiency of distributed computing paradigms such as MPI, e.g., the RAY project (https://github.com/ray-project/ray), which may encourage more genomics applications to take advantage of the distributed systems.

2.4. Cloud Scalability

The need to process an unprecedented amount of genomics data efficiently and robustly also drives many applications built on cloud computing technologies (reviewed in [33]). In cloud computing paradigms, data are distributed to a large number of nodes and computation is shifted to the node where the data resides. Hadoop and Spark are two powerful big-data frameworks for cloud computing.

The Hadoop framework [34] is the Apache's open-source implementation of Google's MapReduce [35] programming model. With the combination of its two core components, Hadoop

Distributed File System (HDFS) [36] and MapReduce, Hadoop enables a load-balanced, scalable, and robust solution for big data analytics. Many Hadoop-based applications have been developed for genomics, to name a few, NGS read alignment [37,38], genetic variant calling [39], sequence analysis [40,41].

The IO-intensive nature of Hadoop's MapReduce can severely limit its performance. For example, map tasks of genomics applications often produce 10–100× amount of intermediate data stored in local disks until the reduce tasks remotely fetch them (pull) via HTTP connections, adding very significant communication overhead. Since there is no state shared between individual map and reduce tasks, Hadoop is not suitable for iterative tasks. Reusing the same dataset across multiple iterations is very common in sequence analysis algorithms, making Hadoop-based solutions very inefficient compared to implementations of MPI and OpenMP.

Apache Spark [42] was developed to overcome the above limitations of Hadoop by providing an efficient abstraction for in-memory computing called Resilient Distributed Dataset (RDD) [43]. Spark can hold the intermediate data and computations in memory as persisted RDDs, thereby improving performance by reducing disk overhead. Besides Scala, Spark also supports other programming languages such as Python and R, which greatly improves its programmability. Integration with the Jupyter notebook interactive environment, or a managed platform provided by Databricks [44], can significantly shorten software development and data analysis time. Many Spark-based genomics applications have been developed for large-scale sequence processing on public or private cloud systems [45–47]; for a comprehensive review, please refer to [45].

Spark with in-memory data processing is significantly faster than Hadoop, but it is still slower than MPI-based implementation, largely owing to the latter's low-level programming language and reduced overhead (e.g., no fault handing like Spark has). Another challenge is that not all the components in a complex genomics data processing pipeline can be easily ported to Spark due to the lack of corresponding libraries.

Cloud-based genomics solutions are evolving in a rapid pace. Cloud systems integrate data management (store, access, and share) and data analysis into one platform, providing flexibility to scale in/out and up/down, and offer user-friendly, consistent, reproducible data pipelines. Such solutions, e.g., Terra (https://app.terra.bio/), will soon unshackle genomics data scientists from the burden of managing hardware and software infrastructures and enable large team collaborations.

2.5. Container Scalability

Bioinformatic pipelines often consist of several independent tools or modules, and these components may require different running environments or have different software dependencies, making them difficult to deploy, manage, and run. Containerization packages all components of a pipeline and their dependencies into a container image so that the pipeline can run consistently on any infrastructure across on-premise and cloud environments. Containers are gaining much popularity because it addresses the challenges of sharing bioinformatics tools and enabling reproducible analyses. For example, ORCA (the genomics Research Container Architecture [48]) provides containers of over 600 bioinformatics tools (as of October 2019). Another resource, Dockerhub (hub.docker.com), also hosts 100+ genomics-related containers. Among the available container platforms, Docker [49] is by far the most popular choice. One major limitation of Docker is its root-owned daemon process, which is not acceptable on most HPC systems. Shifter [50] and Singularity [51] are docker-alternatives that support HPC-based containers.

Containers are "units of deployment". One can host more containers on a single node by upgrading its hardware resources (scale-up). For heavy workloads, Kubernetes [52] (often abbreviated just as "k8s") is a container-orchestration system that scales out container services on a cluster of nodes. It achieves this by abstracting infrastructure components coordination, auto-scaling, and self-healing. Kubernetes works with many container engines and enables automated deployment on cloud or on-premise clusters. One of the successful adoptions of Kubernetes in genomics is the

Galaxy project [53]. However, Kubernetes does not reduce the deployment complexity for applications with complex workflows. To overcome this limit, one solution is to encapsulate a set of Kubernetes resources into a preconfigured package, or charts, and manage the charts using a manager like Helm [54]. Another solution is to use a container-native workflow engine such as Argo [55], which supports both directed acyclic graph (DAG) and step-based workflows on Kubernetes. For deploying machine-learning workflows, Kubeflow [56] makes it simple, portable, and scalable. Despite these efforts, compared to well-established resource managers and schedulers such as slurm or torque in HPC and YARN in the cloud, Kubernetes is still in its infancy.

3. Conclusions and Future Perspectives

We were only able to cover a few scaling strategies for genomics analysis in Section 2, summarized in Table 1 in the context of ease of development, robustness, scalability, and efficiency. In the table, we also listed some bioinformatic tools using these strategies. One might have to combine several strategies, for example, using a single node for development and HPC for large-scale production. When the scale of the analysis exceeds one's on-premise capacity, they can "spillover" to cloud-based solutions for additional capacity.

Table 1. A Comparison of Current Scalable Technologies.

	Shared-Memory Multicore Architecture	**Special Hardware**	**HPC**	**Cloud**	**Container Orchestration**
Ease of dev	+++	+	+	++	+++
Robustness	++	++	+	+++	+++
Scalability	+	+	+++	+++	+++
Efficiency	++	+++	++	+	+
Representative Software	SPAdes [9], OpenMP LCS [11], ALLPATHS-LG [12], SOAPdenovo [13], LSHvec [57]	FAGP [17], SOAP3 [58], SOAP3-dp [59], CUSHAW2-GPU [60] FPGA-based Smith-Waterman [61]	pBWA [27], Ray [28], Meta-HipMer [32] ClustalW-MPI [62] TREE-PUZZLE [63]	HAlign [37] BigBWA [38], BioPig [40,41], SpaRC [46], Metaspark [47], SparkBWA [64]	Galaxy-Kubernetes [53]

It is important to note that genomic analysis pipelines are complex sequential-parallel systems consisting of multiple bioinformatic tools, each with a variable degree of parallelism. The overall scalability is often limited by the step with the least scalability. In addition, the performance gain from increasing parallelism is limited by the Amdahl's law [65] in both parallel and distributed systems.

Serverless, or "Function-as-a-Service", is another interesting concept that is currently undergoing rapid development. It eliminates the need to set up computing infrastructure and brings several benefits including ease-of-use, instantaneous scalability, and cost-effectiveness. Breaking up a gigantic monolithic application into smaller micro-services enables scaling up individual functions, reducing development time, and increasing agility as the pipeline evolves faster. We expect more and more developers to deploy their complex bioinformatics workflows as serverless web services by taking advantage of the rich cloud ecosystems offered by major commercial cloud vendors (AWS Lambda, Microsoft Azure, and Google Clouds Function, etc.)

Author Contributions: L.S. and Z.W. wrote and edited the manuscript.

Funding: Zhong Wang's work was supported by the U.S. Department of Energy Joint Genome Institute, a DOE Office of Science User Facility, under Contract No. DE-AC02-05CH11231.

Acknowledgments: The authors thank Rob Egan and Alex Copeland for their constructive feedback and discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goodwin, S.; McPherson, J.D.; McCombie, W.R. Coming of age: Ten years of next-generation sequencing technologies. *Nat. Rev. Genet.* **2016**, *17*, 333. [CrossRef] [PubMed]
2. Stephens, Z.D.; Lee, S.Y.; Faghri, F.; Campbell, R.H.; Zhai, C.; Efron, M.J.; Iyer, R.; Schatz, M.C.; Sinha, S.; Robinson, G.E. Big data: Astronomical or genomical? *PLoS Biol.* **2015**, *13*, e1002195. [CrossRef] [PubMed]
3. Van Hout, C.V.; Tachmazidou, I.; Backman, J.D.; Hoffman, J.X.; Yi, B.; Pandey, A.; Gonzaga-Jauregui, C.; Khalid, S.; Liu, D.; Banerjee, N.; et al. Whole exome sequencing and characterization of coding variation in 49,960 individuals in the UK Biobank. *bioRxiv* **2019**, 572347. [CrossRef]
4. Papageorgiou, L.; Eleni, P.; Raftopoulou, S.; Mantaiou, M.; Megalooikonomou, V.; Vlachakis, D. Genomic big data hitting the storage bottleneck. *EMBnet J.* **2018**, *24*, e910. [CrossRef]
5. Brian Couger, M.; Pipes, L.; Squina, F.; Prade, R.; Siepel, A.; Palermo, R.; Katze, M.G.; Mason, C.E.; Blood, P.D. Enabling large-scale next-generation sequence assembly with Blacklight. *Concurr. Comput. Pract. Exp.* **2014**, *26*, 2157–2166. [CrossRef]
6. Clavijo, B.J.; Venturini, L.; Schudoma, C.; Accinelli, G.G.; Kaithakottil, G.; Wright, J.; Borrill, P.; Kettleborough, G.; Heavens, D.; Chapman, H.; et al. An improved assembly and annotation of the allohexaploid wheat genome identifies complete families of agronomic genes and provides genomic evidence for chromosomal translocations. *Genome Res.* **2017**, *27*, 885–896. [CrossRef]
7. Wikipedia. POSIX Threads—Wikipedia, The Free Encyclopedia. 2019. Available online: http://en.wikipedia.org/w/index.php?title=POSIX%20Threads&oldid=917571865 (accessed on 29 September 2019).
8. OpenMP. Available online: https://www.openmp.org/ (accessed on 5 December 2019).
9. Bankevich, A.; Nurk, S.; Antipov, D.; Gurevich, A.A.; Dvorkin, M.; Kulikov, A.S.; Lesin, V.M.; Nikolenko, S.I.; Pham, S.; Prjibelski, A.D.; et al. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.* **2012**, *19*, 455–477. [CrossRef]
10. Sathe, S.; Shrimankar, D. Parallelization of DNA sequence alignment using OpenMP. In Proceedings of the 2011 International Conference on Communication, Computing & Security, Odisha, India, 12–14 Februrary 2011; pp. 200–203.
11. Shikder, R.; Thulasiraman, P.; Irani, P.; Hu, P. An OpenMP-based tool for finding longest common subsequence in bioinformatics. *BMC Res. Notes* **2019**, *12*, 220. [CrossRef]
12. Gnerre, S.; MacCallum, I.; Przybylski, D.; Ribeiro, F.J.; Burton, J.N.; Walker, B.J.; Sharpe, T.; Hall, G.; Shea, T.P.; Sykes, S.; et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc. Natl. Acad. Sci. USA* **2011**, *108*, 1513–1518. [CrossRef]
13. Li, R.; Zhu, H.; Ruan, J.; Qian, W.; Fang, X.; Shi, Z.; Li, Y.; Li, S.; Shan, G.; Kristiansen, K.; et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.* **2010**, *20*, 265–272. [CrossRef]
14. Wikipedia. Field-Programmable Gate Array—Wikipedia, The Free Encyclopedia. 2019. Available online: http://en.wikipedia.org/w/index.php?title=Field-programmable%20gate%20array&oldid=916655903 (accessed on 30 September 2019).
15. Wikipedia. Graphics Processing Unit—Wikipedia, The Free Encyclopedia. 2019. Available online: http://en.wikipedia.org/w/index.php?title=Graphics%20processing%20unit&oldid=916685262 (accessed on 30 September 2019).
16. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 24–28 June 2017; pp. 1–12.
17. Flexibility Powered with Acceleration. Available online: https://www.falconcomputing.com/falcon-accelerated-genomics-pipeline/ (accessed on 5 December 2019).
18. 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*, 1297–1303. [CrossRef] [PubMed]
19. Friedrichs, M.S.; Eastman, P.; Vaidyanathan, V.; Houston, M.; Legrand, S.; Beberg, A.L.; Ensign, D.L.; Bruns, C.M.; Pande, V.S. Accelerating molecular dynamic simulation on graphics processing units. *J. Comput. Chem.* **2009**, *30*, 864–872. [CrossRef] [PubMed]

20. Le Grand, S.; Götz, A.W.; Walker, R.C. SPFP: Speed without compromise—A mixed precision model for GPU accelerated molecular dynamics simulations. *Comput. Phys. Commun.* **2013**, *184*, 374–380. [CrossRef]
21. Nobile, M.S.; Cazzaniga, P.; Tangherloni, A.; Besozzi, D. Graphics processing units in bioinformatics, computational biology and systems biology. *Brief. Bioinform.* **2016**, *18*, 870–885. [CrossRef]
22. Webb, S. Deep learning for biology. *Nature* **2018**, *554*, 7693. [CrossRef]
23. AlQuraishi, M. End-to-end differentiable learning of protein structure. *Cell Syst.* **2019**, *8*, 292–301. [CrossRef]
24. The Staggering Cost of Training SOTA AI Models. Available online: https://medium.com/syncedreview/the-staggering-cost-of-training-sota-ai-models-e329e80fa82 (accessed on 5 December 2019).
25. Wikipedia. Message Passing Interface—Wikipedia, The Free Encyclopedia. 2019. Available online: http://en.wikipedia.org/w/index.php?title=Message%20Passing%20Interface&oldid=914967971 (accessed on 29 September 2019).
26. Mallón, D.A.; Taboada, G.L.; Teijeiro, C.; Tourino, J.; Fraguela, B.B.; Gómez, A.; Doallo, R.; Mourino, J.C. Performance evaluation of MPI, UPC and OpenMP on multicore architectures. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 174–184.
27. Peters, D.; Luo, X.; Qiu, K.; Liang, P. Speeding up large-scale next generation sequencing data analysis with pBWA. *J. Appl. Bioinform. Comput. Biol.* **2012**, *1*, 10–4172. [CrossRef]
28. Boisvert, S.; Raymond, F.; Godzaridis, É.; Laviolette, F.; Corbeil, J. Ray Meta: Scalable de novo metagenome assembly and profiling. *Genome Biol.* **2012**, *13*, R122. [CrossRef]
29. Wikipedia. Partitioned Global Address Space—Wikipedia, The Free Encyclopedia. 2019. Available online: http://en.wikipedia.org/w/index.php?title=Partitioned%20global%20address%20space&oldid=865134447 (accessed on 29 September 2019).
30. Berkeley UPC: Unified Parallel C. Available online: https://upc.lbl.gov/ (accessed on 5 December 2019).
31. Zheng, Y.; Kamil, A.; Driscoll, M.B.; Shan, H.; Yelick, K. UPC++: A PGAS extension for C++. In Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, 19–23 May 2014; pp. 1105–1114.
32. Georganas, E.; Egan, R.; Hofmeyr, S.; Goltsman, E.; Arndt, B.; Tritt, A.; Buluç, A.; Oliker, L.; Yelick, K. Extreme scale de novo metagenome assembly. In Proceedings of the SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, 11–16 Novomber 2018; pp. 122–134.
33. Langmead, B.; Schatz, M.C.; Lin, J.; Pop, M.; Salzberg, S.L. Searching for SNPs with cloud computing. *Genome Biol.* **2009**, *10*, R134. [CrossRef]
34. Apache Hadoop. Available online: https://hadoop.apache.org/ (accessed on 5 December 2019).
35. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
36. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The hadoop distributed file system. In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Washington, DC, USA, 3–7 May 2010; Volume 10, pp. 1–10.
37. Zou, Q.; Hu, Q.; Guo, M.; Wang, G. HAlign: Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy. *Bioinformatics* **2015**, *31*, 2475–2481. [CrossRef] [PubMed]
38. Abuín, J.M.; Pichel, J.C.; Pena, T.F.; Amigo, J. BigBWA: Approaching the Burrows—Wheeler aligner to Big Data technologies. *Bioinformatics* **2015**, *31*, 4003–4005. [CrossRef] [PubMed]
39. Hung, C.L.; Lin, Y.L.; Hua, G.J.; Hu, Y.C. CloudTSS: A TagSNP selection approach on cloud computing. In Proceedings of the International Conference on Grid and Distributed Computing, Jeju Island, Korea, 8–10 December 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 525–534.
40. Nordberg, H.; Bhatia, K.; Wang, K.; Wang, Z. BioPig: A Hadoop-based analytic toolkit for large-scale sequence data. *Bioinformatics* **2013**, *29*, 3014–3019. [CrossRef] [PubMed]
41. Shi, L.; Wang, Z.; Yu, W.; Meng, X. A case study of tuning MapReduce for efficient Bioinformatics in the cloud. *Parallel Comput.* **2017**, *61*, 83–95. [CrossRef]
42. Apache Spark. Available online: https://spark.apache.org/ (accessed on 5 December 2019).

43. Zaharia, M.; Chowdhury, M.; Das, T.; Dave, A.; Ma, J.; McCauley, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012; USENIX Association: Berkeley, CA, USA, 2012; p. 2.
44. Databricks Unified Analytics Platform for Genomics. Available online: https://docs.databricks.com/applications/genomics/index.html/ (accessed on 5 December 2019).
45. Guo, R.; Zhao, Y.; Zou, Q.; Fang, X.; Peng, S. Bioinformatics applications on apache spark. *GigaScience* **2018**, *7*, giy098. [CrossRef]
46. Shi, L.; Meng, X.; Tseng, E.; Mascagni, M.; Wang, Z. SpaRC: Scalable sequence clustering using Apache Spark. *Bioinformatics* **2018**, *35*, 760–768. [CrossRef]
47. Zhou, W.; Li, R.; Yuan, S.; Liu, C.; Yao, S.; Luo, J.; Niu, B. Metaspark: A spark-based distributed processing tool to recruit metagenomic reads to reference genomes. *Bioinformatics* **2017**, *33*, 1090–1092. [CrossRef]
48. Jackman, S.D.; Mozgacheva, T.; Chen, S.; O'Huiginn, B.; Bailey, L.; Birol, I.; Jones, S.J. ORCA: A comprehensive bioinformatics container environment for education and research. *Bioinformatics* **2019**, *35*, 4448–4450. [CrossRef]
49. Merkel, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.* **2014**, *2014*, 2.
50. Gerhardt, L.; Bhimji, W.; Fasel, M.; Porter, J.; Mustafa, M.; Jacobsen, D.; Tsulaia, V.; Canon, S. Shifter: Containers for hpc. *J. Phys. Conf. Ser.* **2017**, *898*, 082021. [CrossRef]
51. Kurtzer, G.M.; Sochat, V.; Bauer, M.W. Singularity: Scientific containers for mobility of compute. *PLoS ONE* **2017**, *12*, e0177459. [CrossRef] [PubMed]
52. Wikipedia. Kubernetes—Wikipedia, The Free Encyclopedia. 2019. Available online: http://en.wikipedia.org/w/index.php?title=Kubernetes&oldid=920543477 (accessed on 4 November 2019).
53. Moreno, P.; Pireddu, L.; Roger, P.; Goonasekera, N.; Afgan, E.; Van Den Beek, M.; He, S.; Larsson, A.; Ruttkies, C.; Schober, D.; et al. Galaxy-Kubernetes integration: Scaling bioinformatics workflows in the cloud. *BioRxiv* **2018**, 488643. [CrossRef]
54. Helm: The Package Manager for Kubernetes. Available online: https://github.com/helm/helm (accessed on 5 December 2019).
55. Argo: Workflow Engine for Kubernetes. Available online: https://github.com/argoproj/argo (accessed on 5 December 2019).
56. Kubeflow: The Machine Learning Toolkit for Kubernetes. Available online: https://www.kubeflow.org/ (accessed on 5 December 2019).
57. Shi, L.; Chen, B. A Vector Representation of DNA Sequences Using Locality Sensitive Hashing. *bioRxiv* **2019**, 726729. [CrossRef]
58. Liu, C.M.; Wong, T.; Wu, E.; Luo, R.; Yiu, S.M.; Li, Y.; Wang, B.; Yu, C.; Chu, X.; Zhao, K.; et al. SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads. *Bioinformatics* **2012**, *28*, 878–879. [CrossRef]
59. Luo, R.; Wong, T.; Zhu, J.; Liu, C.M.; Zhu, X.; Wu, E.; Lee, L.K.; Lin, H.; Zhu, W.; Cheung, D.W.; et al. SOAP3-dp: Fast, accurate and sensitive GPU-based short read aligner. *PLoS ONE* **2013**, *8*, e65632. [CrossRef] [PubMed]
60. Liu, Y.; Schmidt, B. CUSHAW2-GPU: Empowering faster gapped short-read alignment using GPU computing. *IEEE Des. Test* **2013**, *31*, 31–39.
61. Li, I.T.; Shum, W.; Truong, K. 160-fold acceleration of the Smith-Waterman algorithm using a field programmable gate array (FPGA). *BMC Bioinform.* **2007**, *8*, 185. [CrossRef]
62. Li, K.B. ClustalW-MPI: ClustalW analysis using distributed and parallel computing. *Bioinformatics* **2003**, *19*, 1585–1586. [CrossRef]
63. Schmidt, H.A.; Strimmer, K.; Vingron, M.; von Haeseler, A. TREE-PUZZLE: Maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics* **2002**, *18*, 502–504. [CrossRef]
64. Abuín, J.M.; Pichel, J.C.; Pena, T.F.; Amigo, J. SparkBWA: Speeding up the alignment of high-throughput DNA sequencing data. *PLoS ONE* **2016**, *11*, e0155461. [CrossRef]
65. Hill, M.D.; Marty, M.R. Amdahl's law in the multicore era. *Computer* **2008**, *41*, 33–38. [CrossRef]