Big Biological Data Management

Edvard Pedersen and Lars Ailo Bongo

Abstract With the deluge of omics data, the life sciences have become a big data science. The management and analysis of omics data share many of the challenges and technical solutions of other big data fields. However, there are also unique challenges. In particular, there is a need for data management solutions that are backward compatible with unmodified tools, but at the same time scales to large-scale datasets, and in addition manages the intermediate, meta-data, and provenance data of analysis pipelines. In this chapter we present and discuss challenges and approaches for such big biological data management.

1 Introduction

The cost of producing data in bioinformatics is rapidly decreasing [39]. This has resulted in several peta-scale omics data repositories [11]. In addition, there is a similar growth in reference databases that contain data analysis results [37]. However, with rapidly increasing dataset sizes, the analysis cost and resource usage is also rapidly increasing. The wealth of data therefore requires new approaches and technical solutions for biological data analysis. In this chapter we will explore challenges and the use of state-of-the-art technical solutions for big biological data analysis with a focus on data management.

The current state-of-the-art in big data management [1] include systems such as Amazon RedShift [18], Google's Dremel [25], Apache Spark [40] and MapReduce [8]. Most of these were developed to analyze text based data with few dimensions. Biological data differs in that it has more dimensions and noise, it is heterogeneous both with regards to biological content and data formats, and the statistical analysis methods are often more complex. It is therefore not straight forward to adapt these state-of-the-art systems for biological data analysis, nor to integrate these with the analysis framework. It is challenging to even know at which level of the data analysis stack to integrate them.

Edvard Pedersen
University of Tromsø- The Arctic University of Norway, e-mail: edvard.pedersen@uit.no

Lars Ailo Bongo
University of Tromsø- The Arctic University of Norway, e-mail: larsab@cs.uit.no

In this chapter, we give an introduction to biological data analysis, with short descriptions of the workflows, pipelines and execution environments used in the field. In addition, we provide a case study from our own lab. Then we provide a short review of big data storage and processing solutions, highlighting advantages and disadvantages of different approaches for biological data management. Finally, we summarize our own experiences in biological data management using big data systems.

2 Biological Data Analysis

In this section we provide the necessary background required to understand the data management requirements for biological data analysis. We describe how biological data analysis is typically implemented, configured, and executed. We use our own META-pipe [31] pipeline from the metagenomics field as a case study.

2.1 Metagenomic data analysis

The typical analysis of metagenomic data involves the following steps:

1. Retrieve and prepare the sample. This includes sample cleaning and DNA isolation.
2. Analyze the sample using instruments such as next-generation sequencing machines.
3. Raw data processing. This is often done using vendor specific software and operating procedures, and it is typically done at the instrument lab.
4. Quality control and data cleaning of the data received from the instrument lab. This is typically the first step done by the researcher, and it can often be done once for each dataset.
5. Run the data through a series of tools in a pipeline to produce the output data needed to answer a particular research question. The same data may be used in several data analysis pipelines, and a pipeline may be run periodically to update the results with new input data or with updated reference databases.
6. Analyze the output data. This is typically done using interactive visual tools that are often decoupled from the analysis pipeline.

In this chapter we will focus on the data analysis pipeline (step 5). This is the step where the researchers put most effort into development time and computation resources.

2.2 Data analysis pipelines

Biological data analysis is typically done through a collection of tools arranged in a pipeline where the output of one tool is the input to the next tool (figure 1). The data transformations include file conversion, data cleaning, normalization, and data integration. A specific biological data analysis project often requires a deep workflow that combines many tools [9]. There are many libraries [16, 14, 35] with hundreds of tools, ranging from small, user-created scripts to large, complex applications [23].

There are four types of data managed for such analysis pipelines:

1. The *input, intermediate, and output* data. These are typically structured files with many samples. The files may range in size from megabytes to terabytes. The pipeline input data is produced by biotechnology instruments such as next-generation sequencing machines, or downloaded from public repositories such as GEO [10] and ENA [22]. The pipeline output files that are typically analyzed using standalone interactive visualization tools.
2. *Contextual data* contains information about the data samples required for data selection and interpretation. This includes information about how, where and when the samples were collected. The contextual data may be used to select the datasets and records to process for a pipeline execution, and by visualization and data exploration tools.
3. *Reference databases* with human or machine curated meta-data extracted from the published literature and from analysis of experimental data [12]. These are used to annotate the data to make it useful for scientists. The reference databases range in size from small collections of annotation data (Swiss-Prot), to peta-scale collections of analyzed samples (European Nucleotide Archive).
4. The *Provenance* required for experiment reproducibility. This includes data lineage information with descriptions of the pipeline tools, their parameters, databases and versions, and machines used in the analysis.

2.3 Pipeline frameworks and execution environments

The analyst specifies, configures, and executes the pipeline using a pipeline framework (a review and taxonomy is provided in [23]). The pipeline framework provides a way of specifying the tools and their parameters, management of data and metadata, and execution of the tools. In addition, a pipeline framework may enable data analysis reproducibility by maintaining provenance data such as the version and parameters of the executed tools. It may also maintain the content of input data files, reference databases, output files, and possibly intermediate data.

A pipeline framework may comprise of a set of scripts run in a specific platform, or a system that maps high-level workflow configuration to executable jobs for many platforms. There are also frameworks that provide an interactive GUI for workflow configuration, and a backend that handles data management and tool execution.

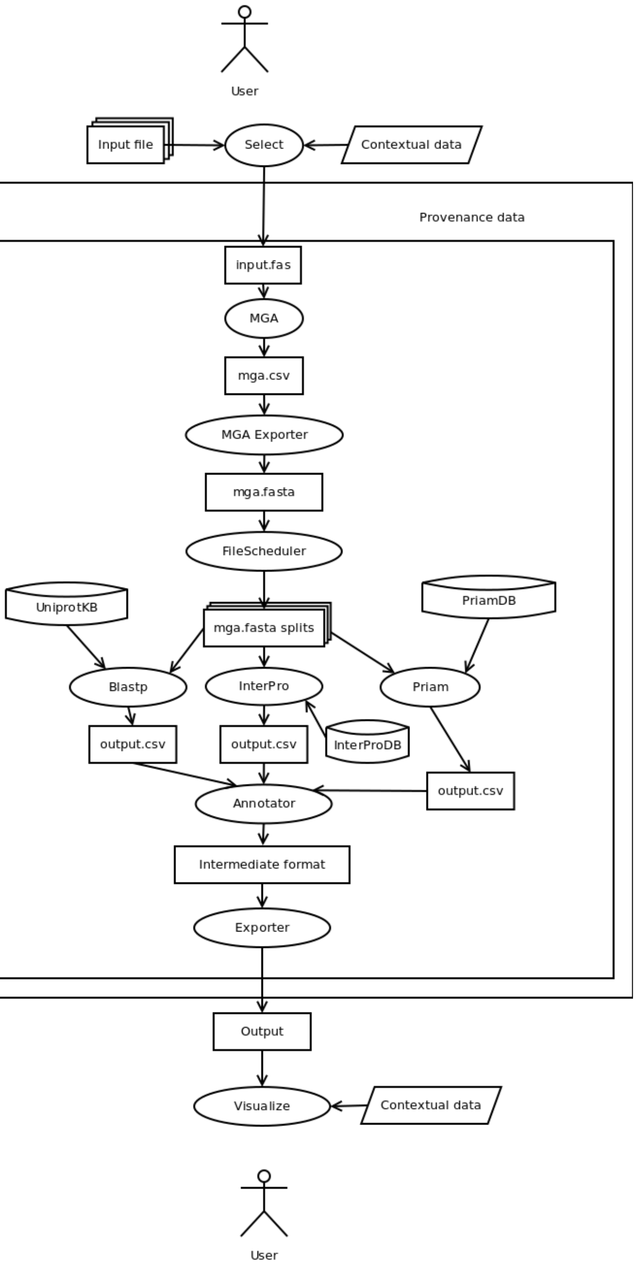

Fig. 1 The META-pipe pipeline tools, file formats, reference databases, and intermediate files. The items inside the "Provenance data" box must have provenance information recorded for analysis reproducability.

Biological data analysis pipelines are typically run on a fat server, high performance computing clusters, or a data-intensive computing cluster. Using a single server has two main advantages. First, most biological analysis tools can be used unmodified. Second, it is not necessary to distribute and maintain tools and data on a cluster. The main disadvantage is the lack of scalability, both concerning dataset size and the number of concurrent users.

Many biological data analysis tools can easily be run on high performance computing (HPC) clusters by splitting the input (or reference databases) into many files that can be computed in parallel. The main advantage of using an HPC cluster is their parallel compute performance. The main disadvantage is that the centralized storage system often becomes a bottleneck for large-scale datasets in I/O bound jobs.

Finally, data-intensive computing clusters [34] distribute storage distributed on the compute nodes, and provide data processing systems that utilize such distributed storage. The main advantage is improved performance and scalability for I/O bound jobs. The main disadvantage is that to fully utilize such a platform the analysis tools may need to be modified [9, 30, 7].

A comparison of how a selection of workflow managers handle different types of data is shown in table 1.

Pipeline	Data	Contextual data	Reference databases	Provenance
Galaxy	Local files	None	Varies	Workflow
GePan	Local files	None	Packaged with pipeline	Partial
EBI Metagenomics	Remote files	Packaged with data	Packaged with tools	Report to user

Table 1 Data management approaches for selected pipeline frameworks.

2.4 Case study: META-pipe

To illuminate the data management issues in bioinformatics, we use as a case study, an in-house workflow manager and pipeline which we have used extensively in research in this field. The pipeline and associated workflow managers are called META-pipe. We are currently developing version 2.0 of the pipeline and workflow manager that will be provided as a European Service in the ELIXIR e-infrastructure.

META-pipe is a successor to GePan, which was a workflow manager designed to do annotation of genomes. META-pipe extends GePan by integrating several new tools, as well as enabling pipelines to be run on supercomputer infrastructure.

META-pipe consists of a workflow manager which automatically generates a pipeline based on the input parameters, and runs this pipeline on the local supercomputer. The pipelines created are for marine metagenomics analysis, and integrates existing biological analysis frameworks with modern data management techniques and infrastructures.

The actual execution in META-pipe works by generating scripts based on the input of the user, these scripts and the input data are then submitted to an execution site, where the reference databases and tools are housed. The tools are run in a data-parallel fashion, enabling the analysis of large samples. The user-facing frontend of META-pipe is Galaxy, which allows users to examine the output of the pipeline in several formats.

Data management in META-pipe is to a large degree up to the pipeline developer and administrator. The input, output, and intermediate data are stored as files managed by the META-pipe scripts. The META-pipe workflow manager handles the intermediate data in the pipeline, but the pipeline developer must maintain the input and output data. For parallel execution, the data must be split and distributed on a cluster by pipeline scripts. Contextual data is not handled by META-pipe, and neither are the reference databases. These must be manually maintained as files stored on a global file system. Finally, some data lineage is provided by the META-pipe job scripts that specify the tools, tool parameters, and file paths. There is however, no automated way of maintaining or specifying file and database versions unless these are encoded in the filenames. We provide additional details in the next chapter.

3 Big Data Management

Biological data analysis jobs have moved from personal computers to data centers and clusters, due to the increased need for storage and data processing resources. However, bioinformatics analysis tools still often rely on files stored locally. It is therefore not straight forward to start using state-of-the-art big data management and processing systems for such analyses, and the developer must consider the strengths and weaknesses of different big data approaches. In this section, we provide an overview of these approaches and systems with respect to biological data management. Throughout the chapter, we will use META-pipe as a case study for how the different approaches may be used.

3.1 Local Data Storage

The most most common interface used by bioinformatics tools is the the file system, where data is stored and managed as files in a directory structure. In addition to files, many tools use simple relational databases to store provenance and contextual data. This approach is sufficient for many cases, where the amount of data is limited. We will not go into details about different databases and file systems in this chapter.

META-pipe version 1.0 uses the file system for data management. This worked well when for small data sets. However, recent flagship metagenomics datasets are too large to be replicated on each compute node. Hence, the data must be stored on a global filesystem, which may then become a performance bottleneck.

3.2 Distributed Data Storage

In computing, it is often necessary to decrease locality to increase capacity. Examples include swapping to disk when a dataset does not fit in memory, or using a multi-level cache. To provide scalable storage and enable large-scale analysis without decreasing locality, the data must be distributed over multiple machines in a way that enables efficient distributed computing. Below we provide some approaches, but note that these approaches are examples of systems which make it easy to develop parallel programs that do computations on local data in a distributed fashion, and hence there are other approaches to distribute data while maintaining locality of data for computations.

One such approach are distributed file systems such as HDFS [34], GPFS [33] and the Google file system [15]. These have been shown to be extendable to large-scale datasets and they have been used to store biological datasets used by genomics analysis pipelines [21, 9, 7] A distributed file system provides programmers with an abstraction of a single file system, while also enabling efficient parallel computations that maintain data locality, such that tasks are scheduled to run on the cluster nodes that contain the data to be processed.

Other big data fields, such as astronomy, distribute their data among multiple standalone relational databases and then use distributed queries in the analysis [38]. There are also distributed databases such as Cassandra [5] and HBase [3], which handle the distribution of queries and data automatically. These approaches both enable locality, either through user defined coroutines, or through frameworks such as Spark or MapReduce. But, we are not aware of bioinformatics tools that extensively use distributed queries.

There are also many other big data management systems that are distributed RDBM systems or NoSQL databases. Systems such as MySQL Cluster [28] are relational databases which are deployed across multiple machines. NoSQL databases often trade off ACID properties for other features to improve for example performance. The NoSQL databases vary from simple in-memory distributed key-value stores such as memcached [13] to almost full-featured databases such as Cassandra[5]. There are also distributed databases that are even more connected than relational databases that are useful for biological data management. For example, graph databases such as Neo4j have been shown to be appropriate for some use cases in bioinformatics. Have et al. [19] used Neo4j to do calculations on protein interactions (which map well to graphs), and acheived a large speedup over PostgreSQL.

For META-pipe, we have used Storage Area Network for storage on the supercomputer, Network Adressed Storage for archiving on our smaller cluster, Network File System for storage on our smaller cluster, HDFS and HBase for expansions like Mario and GeStore, BerkleyDB for the internal representation of reference databases and SQLite as the database for our frontend.

3.3 Generalized Distributed Data Processing

For biological data processing, we are primarily interested in distributed data processing frameworks that: (i) are are easy to use and, (ii) distribute computation efficiently.

Probably the easiest approach for data analysis is to use declarative queries. Many data storage frameworks provide the necessary support for queries for their users. The supported queries includes complex joins supported by relational databases, and simpler operations such as scans and retrieval of a key-value pair supported by key-value stores and NoSQL databases All of these queries enable data locality, provided that the underlying database system supports data locality scheduling of tasks. Many systems also provide support to extend queries with user defined functions (stored procedures) that can be used to implement more complex processing. The combination enables the ease of use, flexibly, and and power to solve many data processing requirements in a distributed fashion.

Many bioinformatics tools and pipelines can utilize graph-based processing frameworks such as MapReduce [8] and Spark [40] (note that these are systems for graph-based processing, and not graph processing systems such as Pregel [24]). These greatly simplifies embarrassingly parallel computations compared to earlier approaches such as MPI. These frameworks automate data distribution, manage data locality concerns, and handle load balancing. These are achieved by partitioning the work into many small data-parallel tasks, which are scheduled and executed by the compute engine of the framework. These frameworks model the computations as a set of shared-nothing operations. For example, the building blocks of the MapReduce framework are a map that transforms all values in a set, and a reduce that performs a summary operation. This allows more flexible processing than is available when using stored procedures or queries, as well as enabling the compute engine to perform optimizations such as efficient load balancing as well as strategies to minimize the effects of stragglers and failures on the parallel program.

The MPI programming model provides operations at an even lower level. The MPI operations are for passing messages between processes running on different nodes in the network, and they thereby allow the developer full control over the data and computational distribution, as well as the communication between nodes. This framework is widely used in bioinformatics tools which support distributed execution. The disadvantage of this approach is that it can be very complex to implement even relatively simple data processing, as all the minutia of data distribution and communication have to be explicitly defined by the developer. The large responsibility put on the developer renders this framework relatively impractical for many types of data processing, where something simpler can be used efficiently.

For META-pipe, the original pipeline was extended with analysis with tools which utilize MPI, as well as batch data processing using MapReduce. The latest version of META-pipe uses Spark extensively.

Approach	Load balance	Data-independent	Complexity	Power
Queries	Yes	No	Low	Low
Stored procedure	Yes	No	Low	Medium
Graph-based processing	Yes	Yes	Medium	High
Message-passing	Yes	Yes	High	High

Table 2 Distributed data processing framework abstractions.

3.4 Specialized data processing

The general purpose frameworks in the previous section have been used to implement specialized data processing systems for uses such as incremental updates, iterative computation, and interactive analysis. These systems provide additional features, besides batch processing and provenance, that are useful for biological data analysis.

Incremental systems, such as the Incoop [6] and Marimba [32] MapReduce extensions, provide iterative computation. These systems reduce the pipeline execution time for updated datasets, by only processing the new data that is appended to a dataset. The results are then combined with previously computed results. For META-pipe, we provide this functionality using GeStore [29], which allows incremental updates for unmodified biological data analysis tools.

Interactive analysis systems, such as Cloudera Impala [20] and Apache PigPen [27], are designed to execute interactive data analysis jobs with very short execution time. Our Mario system, which uses HBase [3] as a backend, is built to interactively tune the parameters of pipeline tools. We have found parameter tuning especially useful for finding the best cutoff value for sample quality in a filtering step. This is a step done early in the pipeline, and parameter changes without Mario would require manually executing the full pipeline for each parameter.

Graph processing systems, such as GraphX [17], are designed for large-scale graph processing. These are used in biological data analysis tools such as in Spaler [2] a de-novo graph-based genome assembler.

4 Big Data Systems for Biological Data Management

To integrate the data management and processing systems described in the previous section with the workflow managers and pipelines in use in bioinformatics, several approaches can be taken:

- Direct integration, where the distributed data management systems are used directly by the workflow manager or pipeline.
- File system integration, where the workflow manager or pipeline is not changed, but instead the file system operations are replaced with use of the data management system through an interface or wrapper.

- Extra tool, where the data management is implemented as a tool in the pipeline, which is used between steps.

We have used all three approaches for integrating our data analysis pipelines with our data processing systems [30].

We have primarily used the Hadoop stack (HDFS, HBase and MapReduce) as the data management and processing backend. We have also researched and tested other systems. We chose these as we had need for several layers of data storage and a simple processing framework. The Hadoop stack provides a simple integrated and mature framework. Using a mix of HBase and HDFS as well as MapReduce enables us to minimize the overhead large-scale data processing expansions that we added to the existing META-pipe workflow manager.

For example, we have been able to achieve a speedup of up to 14x for incremental updates with minimal changes to the workflow system [29]. This speedup comes from generating a small reference database for the tools from a tera-scale collection of reference database versions. In a similar vein, Mario uses HBase to generate tiny workloads, which in turn enables interactive parameter tuning on real data [7].

System	What we have used it for
HDFS	Intermediate files, caching, unstructured data
HBase	Structured and semi-structured data
MapReduce	Computing delta files, parsing and exporting data

Table 3 Data-intensive computing systems we have used in our research.

Other projects that have combined big data systems with biological data analysis includes ADAM [26], which implements an entire variant calling pipeline using Apache Avro [4], Parquet [36] and Spark. They achieve large speedups when compared to traditional tools. They also examine using the same approach with an astronomy workload, where the Spark-based implementation achieved a 8.9x speedup compared to a MPI-based approach.

Diao et al. [9] have used large-scale data processing frameworks to run unmodified bioinformatics tools in parallel. They provide a generalized approach for using these tools and the discuss some of the advantages and challenges of their approach.

For the next version of META-pipe, we plan to use a hybrid of these two approaches, where many of the in-house tools will be implemented in Spark, and the remaining tools will be run unmodified in a data-parallel fashion.

5 Summary

The field of bioinformatics has just recently started to experience the effects of exponential data growth. However, bioinformatics pipelines and tools are typically not designed to scale these large data volumes. There has been large influx of large-scale data processing frameworks that enable efficient distributed data processing in fields such as web-scale data mining and astronomy. By leveraging the knowledge gained in these fields to increase data analysis scalability, the exponential growth of data can be used to increase the quality of bioinformatics analyses.

We have found that integration with legacy pipelines is challenging but it offers great potential to improve analysis performance. Our experiences show that the integration of legacy systems with large-scale data processing can be done without disrupting or supplanting existing pipelines. In addition, it enables features such as better provenance management, data versioning, fault-tolerance, and interactive parameter tuning. We believe our approaches are general and that they can be applied to other data analysis pipelines in bioinformatics and other fields.

References

1. Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A. Bernstein, Michael J. Carey, Surajit Chaudhuri, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J. Franklin, Johannes Gehrke, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Donald Kossmann, Samuel Madden, Sharad Mehrotra, Tova Milo, Jeffrey F. Naughton, Raghu Ramakrishnan, Volker Markl, Christopher Olston, Beng Chin Ooi, Christopher Ré, Dan Suciu, Michael Stonebraker, Todd Walter, and Jennifer Widom. The beckman report on database research. *Commun. ACM*, 59(2):92–99, January 2016.
2. A. Abu-Doleh and V. atalyrek. Spaler: Spark and graphx based de novo genome assembler. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 1013–1018, Oct 2015.
3. Apache. Apache HBase. http://hbase.apache.org. [Online; accessed 18-April-2016].
4. Apache. Avro. http://avro.apache.org. [Online; accessed 18-April-2016].
5. Apache. Cassandra. http://cassandra.apache.org. [Online; accessed 18-April-2016].
6. Pramod Bhatotia, Alexander Wieder, Rodrigo Rodrigues, Umut A Acar, and Rafael Pasquini. Incoop : MapReduce for Incremental Computations. In *Proc. of the 2nd ACM Symposium on Cloud Computing*, page 7. ACM Press, 2011.
7. Lars A Bongo, Edvard Pedersen, and Martin Ernstsen. Data-Intensive Computing Infrastructure Systems for Unmodified Biological Data Analysis Pipelines. In *Computational Intelligence Methods for Bioinformatics and Biostatistics*, volume 8623 of *LNBI*, 2014.
8. Jeffrey Dean and Sanjay Ghemawat. MapReduce. *Communications of the ACM*, 51(1):107, January 2008.
9. Yanlei Diao, Abhishek Roy, and Tony Bloom. Building Highly-Optimized, Low-Latency Pipelines for Genomic Data Analysis. In *Proc. of 7th Biennial Conference on Innovative Data Systems Research*, 2015.
10. Ron Edgar, Michael Domrachev, and Alex E Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.*, 30(1):207–210, 2002.
11. EMBL-European Bioinformatics Institute. EMBL-EBI Annual Scientific Report 2014. http://www.ebi.ac.uk/about/brochures. [Online; accessed 18-April-2016].
12. Xosé M Fernández-Suárez, Daniel J Rigden, and Michael Y Galperin. The 2014 Nucleic Acids Research Database Issue and an updated NAR online Molecular Biology Database Collection. *Nucleic Acids Res.*, 42(Database issue):D1–6, January 2014.
13. Brad Fitzpatrick. Distributed caching with memcached. *Linux J.*, 2004(124):5–, August 2004.
14. Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney,

Jean Y H Yang, and Jianhua Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):R80, January 2004.

15. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.
16. Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86, January 2010.
17. Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 599–613, Broomfield, CO, October 2014. USENIX Association.
18. Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, and Vidhya Srinivasan. Amazon redshift and the case for simpler data warehouses. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1917–1923, New York, NY, USA, 2015. ACM.
19. Christian Theil Have and Lars Juhl Jensen. Are graph databases ready for bioinformatics? *Bioinformatics*, 29(24):3107–3108, 2013, http://bioinformatics.oxfordjournals.org/content/29/24/3107.full.pdf+html.
20. Marcel Kornacker, Alexander Behm, Victor Bittorf, Taras Bobrovytsky, Casey Ching, Alan Choi, Justin Erickson, Martin Grund, Daniel Hecht, Matthew Jacobs, Ishaan Joshi, Lenni Kuff, Dileep Kumar, Alex Leblang, Nong Li, Ippokratis Pandis, Henry Robinson, David Rorke, Silvius Rus, John Russell, Dimitris Tsirogiannis, Skye Wanderman-Milne, and Michael Yoder. Impala: A modern, open-source sql engine for hadoop. In *CIDR*. www.cidrdb.org, 2015.
21. Patricia Kovatch, Anthony Costa, Zachary Giles, Eugene Fluder, Hyung Min Cho, and Svetlana Mazurkova. Big omics data experience. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pages 39:1–39:12, New York, NY, USA, 2015. ACM.
22. Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga, Ying Cheng, Iain Cleland, Nadeem Faruque, Neil Goodgame, Richard Gibson, Gemma Hoad, Mikyung Jang, Nima Pakseresht, Sheila Plaister, Rajesh Radhakrishnan, Kethi Reddy, Siamak Sobhany, Petra Ten Hoopen, Robert Vaughan, Vadim Zalunin, and Guy Cochrane. The European nucleotide archive. *Nucleic Acids Res.*, 39(SUPPL. 1), 2011.
23. Jeremy Leipzig. A review of bioinformatic pipeline frameworks. *Briefings in Bioinformatics*, 2016, http://bib.oxfordjournals.org/content/early/2016/03/23/bib.bbw020.full.pdf+html.
24. Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.
25. Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2):330–339, 2010.
26. Frank Austin Nothaft, Matt Massie, Timothy Danford, Zhao Zhang, Uri Laserson, Carl Yeksigian, Jey Kottalam, Arun Ahuja, Jeff Hammerbacher, Michael Linderman, Michael J. Franklin, Anthony D. Joseph, and David A. Patterson. Rethinking data-intensive science using scalable analytics systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 631–646, New York, NY, USA, 2015. ACM.
27. Christopher Olston, Shubham Chopra, and Utkarsh Srivastava. Generating example data for dataflow programs. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 245–256, New York, NY, USA, 2009. ACM.
28. Oracle. MySQL. http://www.mysql.com. [Online; accessed 18-April-2016].
29. Edvard Pedersen and Lars Ailo Bongo. Large-scale biological meta-database management. *Future Generation Computer Systems*, pages –, 2016.

30. Edvard Pedersen, Inge A Raknes, Martin Ernstsen, and Lars A Bongo. Integrating Data-Intensive Computing Systems with Biological Data Analysis Frameworks. In *Proc. of 23rd Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 733–740. IEEE, 2015.
31. Espen M. Robertsen, Tim Kahlke, Inge A. Raknes, Edvard Pedersen, Erik K. Semb, Martin Ernstsen, Lars A. Bongo, and Nils P. Willassen. Meta-pipe - pipeline annotation, analysis and visualization of marine metagenomic sequence data. 2016, arXiv:1604.04103 [cs.DC].
32. Johannes Schildgen, Thomas Jorg, Manuel Hoffmann, and Stefan Dessloch. Marimba: A Framework for Making MapReduce Jobs Incremental. In *2014 IEEE International Congress on Big Data*, pages 128–135. IEEE, June 2014.
33. Frank Schmuck and Roger Haskin. Gpfs: A shared-disk file system for large computing clusters. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, Berkeley, CA, USA, 2002. USENIX Association.
34. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies*, 0(5):1–10, 2010.
35. Jason E. Stajich, David Block, Kris Boulez, Steven E. Brenner, Stephen A. Chervitz, Chris Dagdigian, Georg Fuellen, James G R Gilbert, Ian Korf, Hilmar Lapp, Heikki Lehväslaiho, Chad Matsalla, Chris J. Mungall, Brian I. Osborne, Matthew R. Pocock, Peter Schattner, Martin Senger, Lincoln D. Stein, Elia Stupka, Mark D. Wilkinson, and Ewan Birney. The Bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12(10):1611–1618, 2002.
36. Twitter, and Cloudera. Parquet. http://www.parquet.io. [Online; accessed 18-April-2016].
37. UniProt Consortium. UniProt release 201504. http://www.uniprot.org/help/2015/04/01/release. [Online; accessed 18-April-2016].
38. Daniel L. Wang, Serge M. Monkewitz, Kian-Tat Lim, and Jacek Becla. Qserv: A distributed shared-nothing database for the lsst catalog. In *State of the Practice Reports*, SC '11, pages 12:1–12:11, New York, NY, USA, 2011. ACM.
39. KA. Wetterstrand. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). http://www.genome.gov/sequencingcosts. [Online; accessed 18-April-2016].
40. Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark : Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, page 10, 2010.