# Data-Intensive Computing Infrastructure Systems for Unmodified Biological Data Analysis Pipelines

Lars Ailo Bongo, Edvard Pedersen, Martin Ernstsen*

Department of Computer Science and Center for Bioinformatics,
University of Tromsø, Tromsø, Norway
*Now at Kongsberg Satellite Services AS, Tromsø, Norway
larsab@cs.uit.no, edvard.pedersen@uit.no,
martin.ernstsen@ksat.no

**Abstract.** Biological data analysis is typically implemented using a deep pipeline that combines a wide array of tools and databases. These pipelines must scale to very large datasets, and consequently require parallel and distributed computing. It is therefore important to choose a hardware platform and underlying data management and processing systems well suited for processing large datasets. There are many infrastructure systems for such data-intensive computing. However, in our experience, most biological data analysis pipelines do not leverage these systems.

We give an overview of data-intensive computing infrastructure systems, and describe how we have leveraged these for: (i) scalable fault-tolerant computing for large-scale biological data; (ii) incremental updates to reduce the resource usage required to update large-scale compendium; and (iii) interactive data analysis and exploration. We provide lessons learned and describe problems we have encountered during development and deployment. We also provide a literature survey on the use of data-intensive computing systems for biological data processing. Our results show how unmodified biological data analysis tools can benefit from infrastructure systems for data-intensive computing.

**Keywords:** data-intensive computing, biological data analysis, flexible pipelines, infrastructure systems.

## 1 Introduction

Recent advances in instrument, computation, and storage technologies have resulted in large amounts of biological data [1]. To realize the full potential for novel scientific insight in the data, it is necessary to transform the data to knowledge through data analysis and interpretation.

Biological data analysis is typically implemented using a deep pipeline that combines a set of tools and databases [2]. Biological data analysis is diverse and specialized, so the pipelines have a wide range of resource requirements. Examples include the 1000 Genomes project [3] with a dataset of 260 TB analyzed on supercomputers and warehouse-scale datacenters; the many databases [4] and web servers [5] built for a specific type of biological process or organism; and simple analyses run using predefined Galaxy pipelines [6]. An important challenge when building data analysis tools and pipelines is therefore to choose a hardware platform and underlying data management and processing systems that satisfies the requirements for a specific data analysis problem.

A data analysis and exploration tool architecture typically has the following components:

- A front-end that provides the *user interface* used by the data analysts, including: web applications, pipeline managers [6], web services [7], and low-level interfaces such as file systems, and cloud APIs [8, 9].
- *Data analysis and interpretation services* including: specialized servers, search engines, R libraries such as BioConductor [10], and tool collections such as Galaxy Toolsheds [11].
- *Infrastructure system for data management* including: file systems, databases, and distributed data storage systems.
- *Infrastructure system for parallel and distributed computing* including: queuing systems [12], and the Hadoop software stack [13].
- *Hardware platform*, such as virtual machines, dedicated servers, small clusters, supercomputers, and clouds.

In this paper, we focus on the choice of data management and processing infrastructure systems. In our experience, most biological data analysis uses a file system in combination with a centralized database for data storage and management, and is run either on a single machine or on a small cluster with a queuing system such as Open Grid Engine [12]. This platform has four main advantages. First, the file system-, and database interfaces are stable, and the technology is reliable. Second, many clusters for scientific computing are designed to use a network file system and to run jobs on a cluster using a queuing system. Third, the developers are familiar with these systems and interfaces. Fourth, there are already hundreds of analysis tools implemented for this platform.

An alternative infrastructure must therefore provide better scalability, performance, or efficiency. Or, it must provide services not available on the standard platform. We give an overview data-intensive computing systems, and describe how these can be leveraged for biological data analysis. We describe how we used these systems to transparently extended flexible biological data analysis frameworks with data intensive computing services. We provide lessons learned, and describe the problems we have encountered during development and deployment of the extended pipelines. In addition, we provide a literature survey on the use of data-intensive computing systems in biological data processing.

Our results make it easier to choose data-intensive computing platforms and infrastructure systems for biological data analysis. First, our systems provide data analysis services not available on the standard platform. Second, they do not require modifying biological analysis tools. We believe this combination is essential to increase the use of data-intensive systems in biological data processing.

## 2 Related Work

There are several specialized infrastructure systems developed for data-intensive computing. Many of these were initially developed and deployed at companies such as Google, Yahoo, Facebook, and Twitter, and then later implemented as open source systems. There are also many new systems under development in academia, the open source community, and the industry. We provide a short description of the features provided by these systems. We limit our description to the most widely used systems and omit many emerging systems, and systems that provide a traditional file system or SQL interface.

Data intensive computing systems are often built on a distributed file systems such as Hadoop Distributed File System (HDFS) [14, 15] that provide reliable storage on commodity component hardware and high aggregate I/O performance. A HDFS cluster co-locates storage and computation resources to avoid the bandwidth bottleneck of transferring large datasets over the network to and from the computation nodes. The main advantage of HDFS for biological data analysis is that the architecture is demonstrated to scale to peta-scale datasets [14, 15], and it is widely used for data-intensive computing in other fields. The main disadvantage is that HDFS does not provide a traditional file system interface, so it is necessary to either rewrite the many data analysis tools that use a POSIX file system interface, to incur an overhead for moving data between HDFS and a local filesystem, or incur the overhead of third-party library such as fuse-dfs [16]. In addition, it is not yet a common platform in scientific computing, so it may be necessary to purchase and build a new cluster with storage distributed on the compute nodes instead of a dedicated storage system.

MapReduce [8, 13] is a widely used programming model and infrastructure system for data-intensive parallel computation. It provides fault-tolerant computation on a HDFS-like file system, and makes it easy to write scalable applications since the system handles data partitioning, scheduling and communication. Biological data analysis applications, especially for next-generation sequencing data, have already been implemented using MapReduce ([17] and [2] provides examples and references). The main advantage of MapReduce is that it scales to peta-scale datasets. In addition, most cloud platforms provide a MapReduce interface. The main disadvantage is that the MapReduce programming model may be too restrictive for some biological applications.

HBase [18, 19] is a column based storage system that provides in-memory caching for low latency random data access, and efficient compression. Biological data analysis applications can use HBase to store data accessed interactively, to implement custom data structures, or to structure data for more efficient compression. Compared to relational databases, HBase does not provide an advanced query engine nor ACID properties. Other systems must implement these on top of HBase if such properties are needed by an application.

An alternative for low-latency query processing is Spark [9, 20]. It offers a richer programming model than MapReduce, including iterative operations. It is well suited to implement machine learning algorithms, and interactive data analysis. Spark uses the Scala programming language, which may be unfamiliar to many developers but bindings exists for Java and Python. Compared to the systems discussed above Spark has just recently become a top-level Apache project, but it is rapidly being adopted by many other open-source and commercial systems.

Several high-level programming models are built on top of MapReduce to make it easier to write data analysis programs, including Pig [21], Hive [22], Cascading [23] and Cassandra [24]. To our knowledge, these are not widely used for biological data analysis (see also the discussion in section 4).

Other data-intensive computing systems for Hadoop, HBase, and Spark includes, Cloudera Impala [25] and Drill [26] that both provide a low-latency SQL query engine (inspired by Dremel [27]), Storm [28] for stream processing, and the Mahout [29] library of machine learning algorithms. To our knowledge, these are also not widely used for biological data analysis (see also the discussion in section 4).

There are many frameworks for specifying and running biological data analysis pipelines. The most widely used systems is Galaxy [6]. It has been integrated with the Hadoop software stack [30].

**Table 1.** Our use of data-intensive computing systems for biological data processing.

| System | Problem | Solution | Issues |
|---|---|---|---|
| **Troilkatt** | Scalable analysis pipelines | HDFS: scalable storage<br>MapReduce: I/O intensive pipeline processing | Memory management |
| **GeStore** | Incremental updates for analysis pipelines | HBase: data structures for generating incremental updates | Hadoop MapReduce job startup time |
| **Mario** | Tuning of analysis pipelines | MapReduce: I/O intensive pipeline processing<br>HBase: sparse data structure, low- latency reads and writes | Performance tuning HBase |

## 3 Transparent Data-Intensive Computing

We have extended several data-intensive computing systems to provide services for biological data analysis. In this section, we answer the following questions:

- Why did we choose a particular infrastructure system?
- What problems did the system solve?
- What are the main limitations of the systems for our use?
- What are the lessons learned during development and deployment?

We have used three clusters, with 5, 10, and 64 nodes, for development and deployment of our systems over a period of five years. All were built for data-intensive computing with storage distributed on the compute nodes. We chose the Hadoop software stack, including HDFS for distributed storage and processing. Hadoop is the mostly used data-intensive computing platform, and it has a very active development community. By using Hadoop, we have benefited from improvements to the infrastructure systems, and the addition of new infrastructure systems such as Spark to the ecosystem. In this section, we describe achievements, issues, and lessons learned.

### 3.1 Troilkatt

Troilkatt is a system for batch processing of large-scale collections of gene expression datasets. We use Troilkatt to process data for the IMP integrated data analysis tool [31]. We built Troilkatt in order to scale our gene expression dataset integration pipeline to process all gene expression datasets for several organisms in NCBI GEO [32]. The pipelines comprise tools for data cleaning, transformation, and signal balancing of these datasets. The integrated data compendium for organisms such as human have ten thousands of datasets. The raw data, pipeline results, and intermediate data in a compendium use tens of terabyte of storage space.

Troilkatt provides infrastructure services for automated genomics compendium management. It consists of five main components (Fig. 1). First, the large genomics compendia maintained by Troilkatt are stored and processed on a cluster. Second, Troilkatt leverages the Hadoop software stack for reliable storage and scalable fault-tolerant data processing. Third, the Troilkatt runtime system provides data management including versioning and a library of tools for downloading and processing data. Fourth, Troilkatt can execute a large collection of external tools and scripts for data processing. Finally, a command line based user interface provides an administrator interface for managing compendium content and steering data processing,

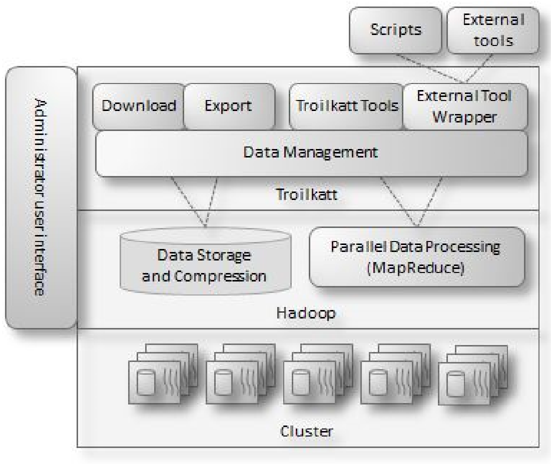

**Fig. 1.** Troilkatt architecture.

The IMP pipeline processing is well suited for data-intensive systems since most pipeline tools are I/O intensive. The data is stored in HDFS. We use MapReduce for parallel processing and HBase for meta-data storage. We chose MapReduce since the processing must scale to several tens of terabytes of data. We use one Mapper task per dataset for each pipeline tool, since the datasets can be processed independently and hence in parallel. In addition, many tools in the IMP pipeline process one row in a gene expression table at a time, and are therefore well suited for the MapReduce programming model.

Initially we considered using Hadoop streaming (MapReduce with unmodified binaries). However, many biological analysis tools require specifying multiple meta-database files as command line arguments, which is not possible with Hadoop Streaming. To solve this problem, Troilkatt allows specifying multiple input files as command line arguments using environment variables.

We achieved a system that efficiently executes pipelines for processing large-scale integrated compendia. We did not have to implement data communication between tasks, nor data locality aware mapping of tasks to compute nodes. Our main issue was large in-memory data structures in two pipeline tools. Hadoop MapReduce is run in a JVM, so the maximum heap size must be set at system startup time. The memory usage of the largest tasks therefore limits the number of tasks that can be run in parallel on each node. To achieve a good trade-off between memory usage and parallelism, we had to divide the expression datasets by their memory usage and processes similarly sized datasets together in a separate MapReduce job.

Since all datasets can be processed in parallel, and most organisms have hundreds or thousands of datasets, the parallelism in the pipeline exceeded the available compute resources even on the biggest 64-node cluster. We can therefore efficiently utilize a much larger cluster. The raw datasets are larger than the available storage on the clusters. We must therefore periodically delete raw data downloaded from public repositories. A full update of the dataset therefore requires re-downloading tens of terabytes of data from public repositories. In addition, we share the clusters with other non-MapReduce jobs. A cluster management system such as Mesos [33] may therefore improve the performance of Troilkatt jobs and improve resource utilization.

## 3.2 GeStore

GeStore [34] is a framework for adding transparent incremental updates to data processing pipelines. We use GeStore to incrementally update large-scale compendia such as the IMP compendia described in the previous section. GeStore is integrated with Troilkatt and Galaxy [35]. We built GeStore since the processing time for a full compendium update can be several days even on a large computer cluster, making it impractical to frequently update large compendia. GeStore adds incremental updates to unmodified pipeline tools by manipulating the input and output files of the tools.

GeStore must detect changes in, and merge updates into, compendia that can be tens of terabytes in size. GeStore must also maintain and generate incremental updates of meta-databases, such as UniProt [36]. These can be hundreds of gigabytes in size. We use HBase for data storage and MapReduce for generating input files and merging output files. HBase provides high-throughput random data accesses required for efficient change detection and merging. GeStore stores meta-database entries as HBase rows, and it updates entries by creating a new version of an HBase table cell. The HBase timestamps enable efficient table scans to find entries that have changed in a period and hence are part of an incremental update. In addition, the flexible schema of HBase tables is utilized to reduce the work required to maintain plugins when file structure or databases change, allowing several years of database versions to be stored in the same HBase table.

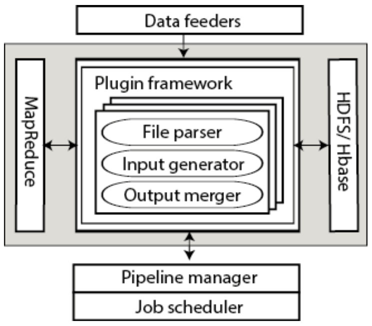

**Fig. 2.** GeStore architecture.

GeStore provides a plugin system to support many biological tools and file formats (Fig. 2). To add incremental updates for a new tool the plugin maintainer implements a plugin. The plugin specifies how to partition input and meta- data files into entries, which part of an entry are required for the analysis done by a tool, and how to compare the entries to detect updates. In addition, the plugin contains code for writing entries to an incremental input file, and merging incremental output with previous output data. These plugins are relatively small in size; less than 300 lines of code in our most complex plugin.

We achieved up to 82% reduction in analysis time for compendium updates when using GeStore with an unmodified biological data analysis pipeline ([34] has additional experimental results). We found HBase to be well suited for the data management requirements of GeStore. File generation and merging scales to large datasets since we use MapReduce for data processing. In addition, we reduce storage space requirements by storing multiple meta-databases versions in HBase instead of storing all versions as separate files.

The overhead introduced by GeStore is high for incremental updates of small datasets, since the startup time of Hadoop MapReduce jobs is tens of seconds. A system with lower startup time, such as Spark, will significantly reduce this overhead. GeStore overhead can also be large if the implemented plugin is not able to properly detect incremental updates.

## 3.3 Mario

Mario [37] is a system for interactive iterative data processing. We have designed Mario for interactive parameter tuning of biological data analysis pipeline tools. For such interactive parameter tuning, the pipeline output should quickly be visible for the pipeline developer. Mario combines reservoir sampling, fine-grained caching of derived datasets, and a data-parallel processing model for quickly computing the results of changes to pipeline parameters. It uses the GeStore approach for transparently adding iterative processing to unmodified data analysis tools.

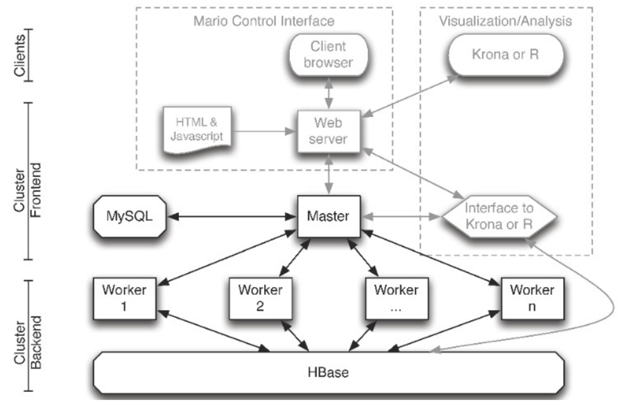

**Fig. 3.** Mario architecture.

The Mario architecture consists of four main components: storage, logic/computation, the web server and the client/UI (figure 3). The system runs on a cluster of computers, with the master process at the cluster frontend, and the workers at the compute nodes. These are co-located with HBase master at the cluster frontend and the HBase region servers at the compute nodes. The web server and the MySQL server can be located on the cluster frontend, or on separate computers. The user runs the Mario controller, and data visualization and analysis tools on her computer.

Mario must efficiently produce random samples from a stream for reservoir sampling, implement a cache of fine-grained pipeline tool results, and implement parallel pipeline stage processing. We use HBase as storage backend due to its low-latency random read and write capability, its ability to efficiently store sparse data structures, and its scalability. Mario stores all intermediate data records produced during pipeline execution in HBase, and uses the cached data to quickly find the data records that must be updated when pipeline tool parameters are changed. Mario also uses HBase for data provenance and single-pass reservoir sampling. The iterative processing in Mario is similar to Spark Streaming [38]. Mario splits the data randomly into many small parts and distributes these on the cluster nodes. Mario process the parts in parallel. However, it only process a small subset at a time since there are many more parts than processor cores. Mario therefore iteratively updates the output data.

We achieved a system for iterative parallel processing that adds less than 100ms of overhead per pipeline stage, and that does not add significant computation, memory, or storage overhead to compute nodes (additional experimental results are in [37]). We found HBase to be very well suited for efficiently storing and accessing the sparse data-structures used by Mario.

Our main issue was to configure HBase to achieve the required performance. We chose a configuration where HBase region servers allocate 12GB DRAM on the cluster nodes, and we traded reliability for improved write latencies by keeping write-ahead logs in memory (and periodically flushing these to disk). The reduced reliability is acceptable for Mario since it can recompute the intermediate data stored in HBase at low cost, if needed.

## 4 Discussion

We have assumed that data-intensive computing systems are not widely used for biological data processing. We base this assumption on our own experience, and discussions we have had with bioinformatics users, model developers, and infrastructure maintainers. In this section, we verify this assumption by conducting a literature survey. We do not conduct a comprehensive survey, but we still believe our results provides insights into the usage of data-intensive computing systems for biological data processing by showing the interest for such systems in the bioinformatics literature. The results for the searches described in this section are in Supplementary materials (http://bdps.cs.uit.no/data/cibb14-supplementary.pdf). [2, 17] provides additional references to articles that describe biological data analysis using data-intensive systems.

### 4.1 Data-Intensive Computing Articles in BMC Bioinformatics

We first examined *Software* articles published in *BMC Bioinformatics*. It is a popular and important journal for bioinformatics analysis tool articles. We also examined *Genome Biology* and the Web Server and Database special issues of *Nucleic Acids Research*, but these journals have few infrastructure articles compared to BMC Bioinformatics. The *Software* articles in that journal also provide detailed implementation details and list of required libraries. We used this information to determine whether the described software uses data-intensive computing systems.

We first got a list of articles from the BMC Bioinformatics website (http://www.biomedcentral.com/bmcbioinformatics/content) by setting *Article* types to *Software*, and *Sections* to *All sections*. We examined the 23 *Software* articles published between August 2014 and November 2014. Of these, two used a HPC platform, one used a data warehouse, and the remaining were implemented for a server or a desktop computer. The results show that most software articles in BMC Bioinformatics describe user interfaces, or data analysis and interpretation services. However, the software in many articles used tools such as BLAST [39] that are ported to both HPC and data-intensive computing platforms. In addition, these articles usually do not describe the backend processing systems used to generate the analyzed data. These may therefore use data-intensive computing systems. To get results that are more specific for our analysis we refined our search to find articles that describe infrastructure systems.

We used the Advanced Search in the Bioinformatics website (http://www.biomedcentral.com/bmcbioinformatics/search). We searched in *All fields* for "infrastructure", and set *Include* to *Software*. This search returned 142 articles in total. We examined 15 articles published between November 2013 and November 2014. Of these, eight described software run on a single server or a desktop computer, six used a cluster for either file storage or as a computational resource, and one system [40] used data-intensive computing techniques. These results indicate that data-intensive computing systems are not widely used for biological data analysis infrastructures.

### 4.2 Articles About Specific Data-Intensive Computing Systems

In this section, we examine articles describing specific data-intensive computing systems. We want to find which systems that are used for biological data processing, and what these systems are used for. We searched for specific keywords and manually filtered the articles in the returned results to exclude articles that do not describe the system we searched for. We set *Include* to *All article types* to increase the number of articles in the search results. The search results includes articles published before December 2014.

We first searched for "Hadoop" since it is the most widely used data-intensive computing platform. The search returned 24 articles. Of these, the software in 14 articles used systems in the Hadoop stack; three articles describe virtual machine images or provisioning systems that include Hadoop, and the remaining only mention Hadoop in related work. The Hadoop systems were used for search, integrated analysis, data integration, machine learning, and distributed analysis of different data types

We also searched for the Hadoop alternative Azure [41]. There were six articles discussing Azure. The software described in two articles used Azure in combination with MapReduce. Two articles propose to use Azure for processing and storage, one describes a virtual machine provisioning systems, and one is a review article.

We then searched for the names of the data-intensive computing systems described in section 2. For *MapReduce* we found 54 articles. In order to remove articles that just mention MapReduce in the citations, we refined the MapReduce search by limiting the search to *Title+Abstract+Text*. This reduced the number of articles to 27. Of these, 12 describe software that used Hadoop MapReduce and three software that used the MapReduce programming model. Most of these articles are also in the Hadoop search results discussed above. We found four *HBase* articles; of which two [40, 42] describe software that used HBase as a storage backend for sequencing data. The remaining two mention HBase in related work. We found two *Spark* articles. One described how they implemented distributed processing of next-generation sequencing data [43] using Spark. *Cassandra*, *Hive*, and *Mahout* were not used by the software in any articles in the search results, but are discussed in the related work in two articles. We did not find *Impala* in any articles.

We also searched for the *Pig*, *Cascading Drill*, and *Storm* systems, but these return many false positives. We therefore refined the search to: "Pig Apache", "Cascading

SQL", "Drill Apache", and "Storm Apache". *Pig* and *Cascading* are both discussed in the related work in two articles. We did not find any articles for *Drill* and *Storm*.

The above results show that MapReduce is the most popular system for data-intensive biological data processing, and that most MapReduce tool implementations use the Hadoop stack. We also found a few articles where HBase and Spark were used. MapReduce, Spark, and HBase are all core systems. We did not find any Bioinformatics articles that described tools that use higher-level systems. This may suggest that the services and abstractions provided by these are not well suited for biological data. The systems that were not mentioned in any articles (Impala, Drill, and Storm) are all recently developed, and may therefore have been unstable when the bioinformatics tools described in the articles were implemented.

### 4.3 Usage Trends

We also examined whether data-intensive computing systems are becoming more used for biological data analysis. To answer this question we counted the number of articles mentioning MapReduce and Hadoop. The MapReduce design [44] was published in 2004, the open-source Hadoop MapReduce project [13] was started in 2005, and Hadoop became popular around 2008. Since then, an increasingly number of articles mention MapReduce and Hadoop each year, especially in the last two years (Table 2). The results indicate that data-intensive computing systems are becoming more popular for biological data processing, but that it takes a few years from the systems become popular until bioinformatics tools use them.

**Table 2.** Number of articles per year for keywords MapReduce and Hadoop (many articles are in both results). 2014 does not include articles published in November and December.

| Year | Hadoop | MapReduce |
|---|---|---|
| 2009 | 1 | 1 |
| 2010 | 4 | 7 |
| 2011 | 4 | 7 |
| 2012 | 4 | 8 |
| 2013 | 4 | 15 |
| 2014 | 7 | 14 |
| **Total** | 24 | 52 |

## 5 Conclusion

We have transparently extended flexible biological data analysis frameworks to utilize data intensive computing infrastructure systems. Our results show that even unmodified biological data analysis tools can benefit from these for: (i) scalable fault-tolerant computing for large-scale data; (ii) incremental updates in order to reduce the resource usage required to update large-scale data compendium; and (iii) interactive data analysis and exploration.

We have identified several limitations of the infrastructure systems we used. However, by using new systems recently added to the Hadoop ecosystem we can remove many of these limitations. We will therefore continue using these systems to extend biological data processing pipelines with new services for making data analysis more efficient and for improving the quality of the analysis results.

We believe that centralized storage system performance is becoming a bottleneck for large-scale biological data processing, and that it will become necessary to use data-intensive computing infrastructure systems and a platform with distributed storage to avoid this bottleneck. The systems presented in this paper motivate and make the transition to such platforms and infrastructure systems easier for two reasons. First, they provide data analysis services that are not available on the standard HPC platform. Second, it is not necessary to modify the biological analysis tools. We believe the combination of novel services and backward compability is essential to increase the use of data-intensive systems in biological data processing.

Troilkatt, GeStore and Mario are all open source:

- https://github.com/larsab/troilkatt
- https://github.com/EdvardPedersen/GeStore
- http://bdps.cs.uit.no/code/mario/

## Acknowledgements

Thanks to Kai Li, Olga Troyanskaya, and Alicja Tadych for their help developing and deploying Troilkatt. Thanks to our colleagues at the Tromsø ELIXIR node for their expertise in developing and deploying biological data processing systems. Thanks to Einar Holsbø, Giacomo Tartari and Bjørn Fjukstad for their comments and insights while writing this paper.

## References

1. Kahn, S.D.: On the Future of Genomic Data. Science (80-. ). 331, 728–729 (2011).
2. Diao, Y., Roy, A., Bloom, T.: Building Highly-Optimized, Low-Latency Pipelines for Genomic Data Analysis. 7th Biennial Conference on Innovative Data Systems Research (CIDR'15), Asilomar, CA, USA (2015).
3. Clarke, L., Zheng-Bradley, X., Smith, R., Kulesha, E., Xiao, C., Toneva, I., Vaughan, B., Preuss, D., Leinonen, R., Shumway, M., Sherry, S., Flicek, P.: The 1000 Genomes Project: data management and community access. Nat. Methods. 9, 459–62 (2012).
4. Fernández-Suárez, X.M., Rigden, D.J., Galperin, M.Y.: The 2014 Nucleic Acids Research Database Issue and an updated NAR online Molecular Biology Database Collection. Nucleic Acids Res. 42, (2014).
5. Benson, G.: Editorial: Nucleic Acids Research annual Web Server Issue in 2014. Nucleic Acids Res. 42, W1–W2 (2014).
6. Goecks, J., Nekrutenko, A., Taylor, J.: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 11, R86 (2010).

7. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics. 20, 3045–54 (2004).
8. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Commun. ACM. 53, 72 (2010).
9. 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. Proc. of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association (2012).
10. Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y.H., Zhang, J.: Bioconductor: open software development for computational biology and bioinformatics. Genome Biol. 5, (2004).
11. Blankenberg, D., Von Kuster, G., Bouvier, E., Baker, D., Afgan, E., Stoler, N., Taylor, J., Nekrutenko, A.: Dissemination of scientific software with Galaxy ToolShed. Genome Biol. 15, 403 (2014).
12. Open Grid Scheduler, http://gridscheduler.sourceforge.net/.
13. Hadoop homepage, http://hadoop.apache.org/.
14. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. 26th Symposium on Mass Storage Systems and Technologies. IEEE (2010).
15. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. ACM SIGOPS Operating Systems Review. p. 29 (2003).
16. MountableHDFS, http://wiki.apache.org/hadoop/MountableHDFS.
17. Taylor, R.C.: An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics. 11 (2010).
18. Apache HBase, http://hbase.apache.org/.
19. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: BigTable: A Distributed Storage System for Structured Data. ACM Trans. Comput. Syst. 26, 1–26 (2008).
20. Apache Spark, https://spark.apache.org/.
21. Gates, A.F., Natkovich, O., Chopra, S., Kamath, P., Narayanamurthy, S.M., Olston, C., Reed, B., Srinivasan, S., Srivastava, U.: Building a high-level dataflow system on top of Map-Reduce: the Pig experience. Proc. of the VLDB Endowment. pp. 1414–1425 (2009).
22. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. Proc. of VLDB Endowment. pp. 1626–1629 (2009).
23. Cascading, http://www.cascading.org/.
24. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Oper. Syst. Rev. 44, 35 (2010).
25. Impala, http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html.
26. Apache Drill, http://incubator.apache.org/drill/.
27. Melnik, S., Gubarev, A., Long, J.J., Romer, G., Shivakumar, S., Tolton, M., Vassilakis, T.: Dremel: interactive analysis of web-scale datasets. Proc. VLDB Endow. pp. 330–339. (2010).
28. Storm, https://storm.incubator.apache.org/.
29. Mahout homepage, https://mahout.apache.org/.

30. Pireddu, L., Leo, S., Soranzo, N., Zanetti, G.: A Hadoop-Galaxy adapter for user-friendly and scalable data-intensive bioinformatics in Galaxy. Proc. of 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics. pp. 184–191. (2014).
31. Wong, A.K., Park, C.Y., Greene, C.S., Bongo, L.A., Guan, Y., Troyanskaya, O.G.: IMP: a multi-species functional genomics portal for integration, visualization and prediction of protein functions and networks. Nucleic Acids Res. 40, W484–90 (2012).
32. Barrett, T., Troup, D.B., Wilhite, S.E., Ledoux, P., Evangelista, C., Kim, I.F., Tomashevsky, M., Marshall, K.A., Phillippy, K.H., Sherman, P.M., Muertter, R.N., Holko, M., Ayanbule, O., Yefanov, A., Soboleva, A.: NCBI GEO: archive for functional genomics data sets--10 years on. Nucleic Acids Res. 39, D1005–10 (2010).
33. Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R., Shenker, S., Stoica, I.: Mesos: a platform for fine-grained resource sharing in the data center. Proc.of the 8th USENIX conference on Networked systems design and implementation. USENIX Association (2011).
34. Pedersen, E., Willassen, N.P., Bongo, L.A.: Transparent incremental updates for Genomics Data Analysis Pipelines. HiBB 2013 – 4th Workshop on High Performance Bioinformatics and Biomedicine. LNCS, vol. 8374. Springer LNCS (2014).
35. Pedersen, E., Raknes, I.A., Ernstsen, M., Bongo, L.A.: Integrating Data-Intensive Computing Systems with Biological Data Processing Frameworks. Euromicro Conference on Parallel, Distributed and Network-Based Processing (2015).
36. Magrane, M., Consortium, U.: UniProt Knowledgebase: a hub of integrated protein data. Database (Oxford). 2011, bar009 (2011).
37. Ernstsen, M., Kjærner-Semb, E., Willassen, N.P., Bongo, L.A.: Mario: interactive tuning of biological analysis pipelines using iterative processing. in Proc. of 5th International Workshop on High Performance Bioinformatics and Biomedicine. LNCS, vol. 8805 (2014).
38. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I.: Discretized streams. Proc. of Twenty-Fourth ACM Symposium on Operating Systems Principles. pp. 423–438. ACM Press (2013).
39. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215, 403–10 (1990).
40. Killcoyne, S., del Sol, A.: FIGG: simulating populations of whole genome sequences for heterogeneous data analyses. BMC Bioinformatics. 15, 149 (2014).
41. Azure: Microsoft’s Cloud Platform, http://azure.microsoft.com/en-us/.
42. O’Connor, B.D., Merriman, B., Nelson, S.F.: SeqWare Query Engine: storing and searching sequence data in the cloud. BMC Bioinformatics. 11 Suppl 1, S2 (2010).
43. Roberts, A., Feng, H., Pachter, L.: Fragment assignment in the cloud with eXpress-D. BMC Bioinformatics. 14, 358 (2013).
44. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Proc. of Operating Systems Design & Implementation. USENIX (2004).