International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-3, Issue-11)

Research Article | **November 2014**

Challenges Involved in High-Throughput Genomics

Sushma R. Vhatkar, Sanchika A. Bajpai
Department Of Computer Engineering
BSIOTR, Pune, India

Abstract—

The flood of data needs to analyse complex processing that mines biological information from vast sets of small sequence reads while handling numerous errors in the data. The development of high-throughput gene sequencing instruments makes mass genomics feasible, but at the same time produces gigabytes of sequencing data per experiment that need to be stored, process, aligned and analyzed. The current approach to data management for high-throughput genomics is file-centric and involves a large number of separate files containing a text-format or a proprietary binary format. Most of these formats are insufficiently documented and do not include metadata. On the other hand, traditional database technology also has shortcomings that prevent it from becoming commonly used in this scientific domain: Database systems are optimized for fast processing of small and precise business data records. In this paper, we are interested to find out whether we have achieved high-throughput and real-time analysis of Genome data using In-Memory technology. We study different scenarios from high-throughput genomics and investigate how current in-memory database technology can be used for efficient data management. In particular, we are interested in faster, fault tolerant, highly available and scalable application.

Keywords—File-centric approach; Genome Data; High-Throughput Genomics; In-Memory Database; Sequence alignment

I. INTRODUCTION

Genomics Genetic analysis labs are facing a serious data management problem: The development of high-throughput gene sequencing instruments makes mass genomics feasible, but at the same time produces gigabytes of sequencing data per experiment that need to be stored, aligned and analyzed. While it took 10 years and $3B dollars to produce a first draft of the human reference genome, the current generation of sequencing instruments is able to sequence between 2 to 4 billion bases in only a few days [1][2]. The current approach to data management for high-throughput genomics is file-centric and involves a large number of separate files containing a text-format or a proprietary binary format. Most of these formats are insufficiently documented and do not include meta-data. Some file formats that include metadata do not separate the data's representation from its conceptual data model. This makes data workflow management very complicated and the analysis becomes inefficient. The current file-centric approach simply does not scale to the terabyte needs of high throughput genomics.

On the other hand, traditional database technology also has shortcomings that prevent it from becoming commonly used in this scientific domain: Database systems are optimized for fast processing of small and precise business data records. Everything beyond these core assumptions is a challenge for current database engines. This includes large BLOB-style attributes – such as gene sequences –, uncertain data – such as most scientific data –, computational intensive data processing functions – such as sequence alignment algorithms –, and efficient support for data provenance and annotation management. But perhaps most importantly, database systems are not easy to deploy and use by the average scientist. They not only require some specific technical skills, but also a certain way of thinking about data design and declarative data access which apparently is not widely embraced by the scientific community.

Foundation for comprehensive genome analysis is DNA sequencing that makes the genetic information encoded in genomes readable. In recent years, next generation sequencing techniques were developed, which sequence complete human genomes in several days [3]. The improvements in DNA sequencing facilitate new use cases for genome analysis such as personalized medicine that tailors disease treatment and drugs towards patients' genomes in order to enable tailor made treatment for patients [4]. Thus, more and more genome sequencing data is generated in even shorter time that must be stored, integrated, processed, and analyzed. In this paper, we contribute a summary of data-management challenges in genome analysis, which must be met to keep pace with DNA sequencing, but also to enable effective use of genome sequencing data. Moreover, we characterize different approaches for comprehensive genome analysis systems regarding their applicability to meet these challenges and discuss the use of main-memory database technologies as basis for such genome analysis systems.

II. RESEARCH BACKGROUND AND RELATED WORK

The Personalized medicine requires comprehensibility and reliability throughout the complete genome-analysis process. This is especially required because of the omnipresent data and result uncertainties within the genome analysis [6]. If genome analysis is not comprehensible and reliable, medical assessments based on genome-analysis results are not transparent and trustworthy and the benefits of personalized medicine will be limited. Using data-management capabilities, such as data integrity, data security, user management, and data provenance, is the foundation for a

© 2014, IJERMT All Rights Reserved

comprehensible and reliable genome-analysis process. Data-integrity and data-security capabilities ensure reliable analysis by keeping data consistent and valid at any time. Data-provenance capabilities allow to track what data contributed how to a certain analysis result. Moreover, user management features allow to define roles and responsibilities of users within the complete genome analysis process making analyses more transparent and comprehensible. Flat-file based command-line tools and distributed processing approaches for genome analysis are not designed for comprehensive data management, but for performance.

In contrast, DBMSs are designed to provide comprehensive data-management capabilities. Thus, matured mechanisms for integrity control, data security, and user management exist. Moreover, DBMSs can be effectively used to provide data-provenance capabilities [7]. Nevertheless, currently, DBMSs are either used as central data storage for external tools or for downstream analysis to facilitate data integration. To provide a holistic approach that guarantees analysis comprehensibility and reliability throughout the complete genome analysis process, an integration of all steps of genome analysis into a DBMS is required.

Sequence alignment and variant calling are the most processing intensive tasks in genome analysis. The runtime of sequence alignment and variant calling depends on the number of reads to align and the size of the reference genome. Next generation sequencing techniques can sequence large genomes in several days and generate large amounts of reads that must be aligned [8]. In contrast, current DNA sequencers have already a throughput of 2.5 billion base pairs per hour [8]. In order to process the increasing amounts of sequencing data efficiently, approaches were developed that use distributed processing frameworks such as MapReduce to speed up the processing. But with increasing amounts of data to process, data transfer times to cloud environments become significant.

In order to reveal new insights on genome mutations and their impacts on organisms, it is necessary to enrich genome sequencing data with further information [3]. Thereby, the required information is stored in many heterogeneous data sources. Thus, matured data-integration capabilities are needed to integrate these heterogeneous data sources making them accessible for comprehensive downstream analysis. Many users participate in analyses and access the database for various purposes. Thus, analyses must not only be fast, but also the interface. The complexity and integration effort of such approaches increases with a higher number of supported tools. Thus, extending such approaches, if possible, is associated with high implementation efforts. Other approaches use traditional disk-based DBMSs to store data from different data sources in a homogeneous data schema, but are designed for specific use cases and provide specialized interfaces beside the standard SQL interface. To the best of our knowledge, there is no evaluation of performance and extensibility of these genome-analysis approaches.

III. SYSTEM OVERVIEW

Our system for genome data processing and analysis works in an environment shown in Fig 1. Users of the system are hospitals and research institutes. A user starts by sending DNA samples to a third-party sequencing service and requests the produced genomic data to be transferred to our system, as shown by the arrow (1) in the figure. For each DNA sample, a sequencer produces raw images and then converts the image data to short read sequences (or reads, or brevity) of the genome. The read sequences are transferred to our system by shipping hard disks, as shown by the arrow (2). The data volume is usually hundreds of gigabytes per genome sample.

Once the data arrives at our system, the user can issue a request to process the data, including alignment of read sequences and detection of variations against a reference genome, as shown by the shaded box labeled as “II. Data Processing”. The output of this module, including a whole genome sequence and variations detected, are stored in a database for further analysis. Afterwards the user can upload additional patient information, and initiate extensive analyses that combine genomic data and patient data. Such analyses are handled by the module labeled as “III. Deep Analysis”, which automatically discovers patterns of both statistical significance and biological meanings.

A. Architecture

As an overall system architecture perspective, our research prototype consists of the architectural layers: data, platform, and application.

1) Data Layer: The data layer holds genomic reference data, such as human reference genomes and annotations [7]. These data is the base for analysis of specific genomic findings. Additionally, it holds the patient-specific genomic data, which was generated by NGS devices. The latter needs to be analyzed in the course of personalized medicine and will be processed by the platform layer and combined by the applications of the application layer.

2) Platform Layer: The platform layer holds the complete process logic and the IMDB system for enabling real-time analysis of genomic data. In Fig. 1 on the right, our developed extensions of the platform layer, the worker and updater framework, are exemplarily depicted. The worker framework specifies for incoming sequencing request required tasks and subtasks and its order comparable to the map reduce approach [8]. It also dispatches these tasks to computing resource, such as computing odes, observes their status, and combines partial result sets to obtain the final result set. The updater framework is the basis for combining international research results. It regularly checks Internet ources, such as public FTP servers or web pages, for updated and newly added annotations, e.g. database exports or characteristic file formats, such as CSV, TXT, etc. New data is automatically downloaded and imported in the IMDB to extend the knowledge base. Once new data was imported, it is available for real-time analysis of genome data without any latency.

3) Application Layer: The application layer consists of special purpose applications to answer medical questions instead of generic purpose applications. Although these applications can only be used for a limited usecase, they are highly optimized for solving these very specific tasks. All applications communicate via asynchronous Ajax calls and

© 2014, IJERMT All Rights Reserved

JavaScript Object Notation as data exchange format via a web service interface with the database layer [9], [10]. Accessing results or performing specific analysis is no longer limited o a single location, e.g. the desktop computer in the office of the physician. All application operations can be accessed from any device configured to have Internet access, which enhances productivity of its users.

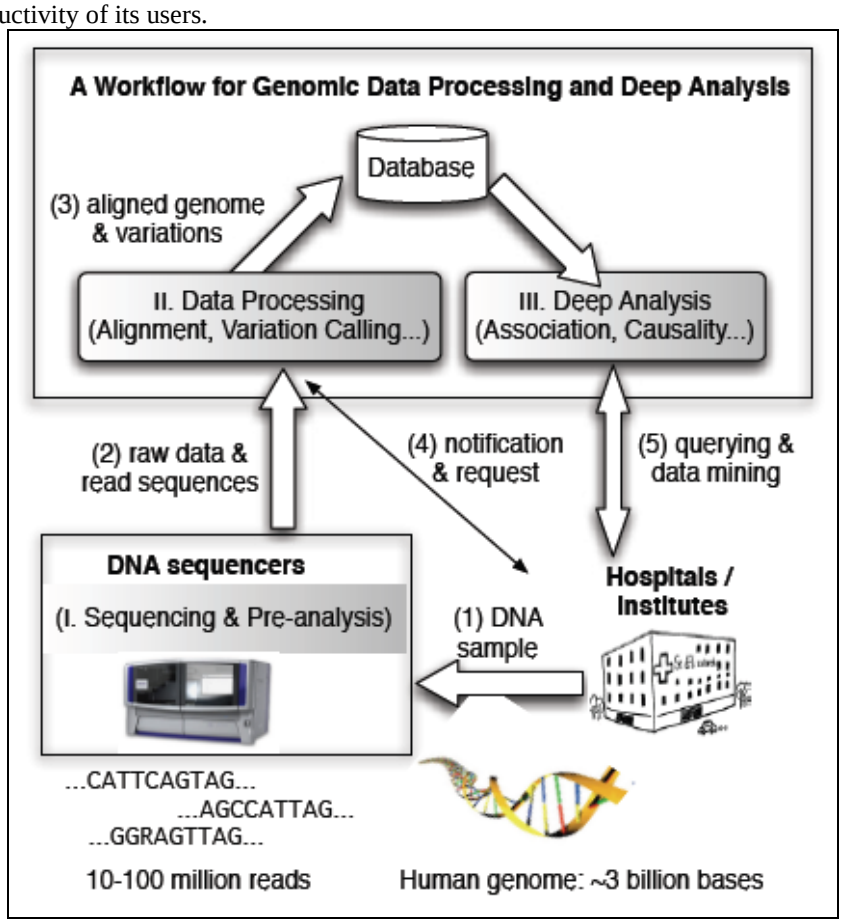

Fig. 1 Overall architecture.

B. Data Modeling

As High-density tiling arrays and new sequencing technologies are generating rapidly increasing volumes of DNA interaction data. Visualization and exploration of this data is critical to understanding the regulatory logic encoded in the genome by which the cell dynamically affects its physiology and interacts with its environment. The Genome Browser is a cross-platform desktop program for interactively visualizing high-throughput data in the context of the genome. Important features include dynamic panning and zooming, keyword search and open interoperability. Users may bookmark locations on the genome with descriptive annotations and share these bookmarks with other users. The program handles large sets of user-generated data using an Oracle Coherence in-memory database.

A dataset has a set of sequences and a set of tracks. Sequences as shown in fig 2, are the chromosomes and plasmids that make up the genome, determine the coordinate system. Tracks shown in fig 3, hold data to be plotted on those coordinates. Track data consists of a set of features, where a feature is a data point related to some location on the genome. To allow for different kinds of features, we store the features for each track in a separate table - one feature table per track. The feature tables are linked to their tracks by the table_name field in the tracks table. All entities identified by a UUID (datasets, sequences and tracks) can be assigned attributes. As shown in fig 4, these are key/value pairs where the value can be a string, integer, floating point number or boolean value.

ID	UUID	Name	Length	Topology
1	213faf4a-e763-4acc-8d2f-d61d4694e23a	chromosome	2014239	circular
2	2df49fe5-8b7a-4623-9c9a-6bef06e712e1	pNRC200	365425	circular
3	a859ef0f-6815-4a84-bbbd-84508736dde4	pNRC100	191346	circular
...				

Fig. 2 Sequence Data sample

© 2014, IJERMT All Rights Reserved

UUID	Name	Type	table_name
22e5d4ba-a1c0-4a9f-921d-53db6e8be038	Genes	gene	features_genes
6206eaff-b0c1-4dc1-8cef-521cbb2dc0a3	Transcript Signal	quantitative.segment	features_transcript_sig
...			

Fig. 3 Track Data sample

UUID	Key	Value
6206eaff-b0c1-4dc1-8cef-521cbb2dc0a3	top	0.37
6206eaff-b0c1-4dc1-8cef-521cbb2dc0a3	height	0.11
6206eaff-b0c1-4dc1-8cef-521cbb2dc0a3	color	0x80336699
6206eaff-b0c1-4dc1-8cef-521cbb2dc0a3	viewer	Scaling
...		

Fig. 4 Attribute Data sample

C. Algorithm Development

In the fig 5, we outline selected building blocks of in-memory computing and their relevance for real-time analysis of genomic data in the context of our work. Lightweight compression techniques refer to a data storage representation that consumes less space than its original pendant [9]. A columnar database storage layout, as used in IMDBs, supports lightweight compression techniques, such as run-length encoding, dictionary encoding, and difference encoding [11]. However, IMDBs automatically perform lightweight compression optimized for the specific data to store. As a result, there is no longer an explicit need to map data from a human-readable format to an optimal storage representation since it is done transparently by the IMDB. Thus, the time to create new applications is reduced, the maintainability of the application code is improved since the source code is easier to understand, and any data stored in the database benefits from this kind of optimization without the need for explicit consideration in the application's code by the software developer. As a result, the database executes all operations on compressed data without the need for explicit decompression, which improves cache-hit ratio since more compressed data fits into the same amount of cache memory [19].

Parallel Data Processing, Latest computer systems consist of multiple cores per individual Central Processing Unit (CPU), which is referred to as multi-core architecture [12]. Additionally, a single server system can be equipped with multiple CPUs multiplying the amount of available computing cores, which is referred to as multi- CPU architecture. The hardware of a single computer system is designed to perform multiple processing tasks simultaneously. However, to use all available computing resources most efficiently software tends to incorporate specific instructions to explicitly make use of parallelization features, e.g., when adding up multiple values using the Parallel Addition (PADD) instruction [13]. Parallelization can be applied to various locations within the application stack of software systems – from within the application running on an application server to query execution in the database system. For example, multiple clinical departments access the data of a single patient simultaneously. Processing multiple queries can be handled by multi-threaded applications, i.e., the application does not stall when dealing with more than one query at the same time. OS threads are a software abstraction that needs to be mapped to physically available hardware resources.

A CPU core is comparable to a single worker on a construction area-[14][15]. If it is possible to map each query to a single core, the system's response time is optimal. Query processing also involves data parallelization, i.e., the database needs to be queried in parallel, too. If the database is able to distribute the workload across multiple cores, a single server works optimal. If the workload exceeds physical capacities of a single system, multiple servers or blades need to be installed for distribution of work to reach optimal processing behavior. From the database point of view, data partitioning supports parallelization since multiple CPU cores even on multiple servers can process data simultaneously. This example shows that multi-core architectures and parallelization depend on each other while data partitioning forms the basis for parallel data processing.

Data Partitioning, We distinguish between vertical and horizontal data partitioning. Vertical partitioning refers to rearranging individual database columns. It is achieved by splitting columns of a database table in two or more sets of columns. Each of the sets can be distributed individually, e.g., on separate databases servers[16][17]. This technique can also be used to maintain the same database column in different ordering to achieve better search performance for mixed workloads while guaranteeing high-availability of data. Key to success of vertical partitioning is a thorough understanding of data access patterns. Attributes that are accessed in the same query should be located in the same partition since identifying and joining additional columns result in additional query processing overhead. In contrast, horizontal partitioning addresses long database tables and how to divide them into smaller chunks of data. As a result, each portion of the database table contains a disjoint subset of the complete data. Splitting data into equivalent long

© 2014, IJERMT All Rights Reserved

horizontal partitions is used to support parallel search operations across all data of a database table and to improve scalability [19]. Applying a horizontal partition per chromosome for the genome table enables scanning of all chromosomes in parallel. Furthermore, applying horizontal partitioning to each of the chromosome database tables enables processing of each individual chromosome by individual CPU resources, e.g., CPU cores.

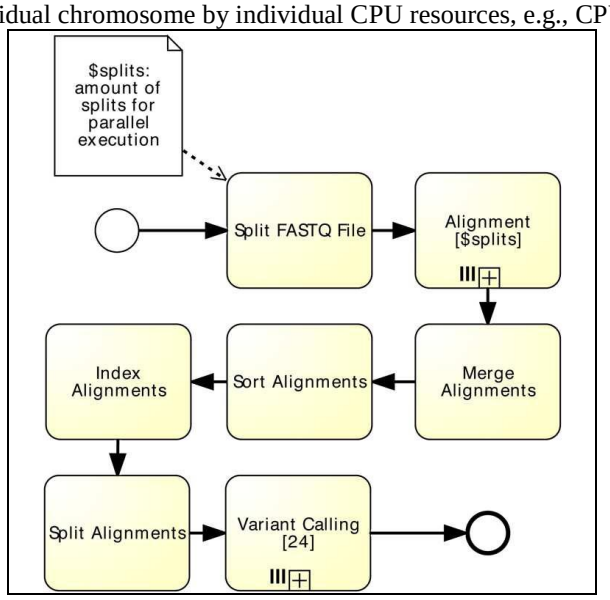

Fig. 5 Variant calling is processed in parallel

As shown in fig 6, the inputs for alignment tasks are FASTQ files containing thousands or millions of raw DNA reads or snippets. FASTQ files are generated by the NGS device in a time intensive process. Instead of waiting for a single huge FASTQ file, we start processing as soon as possible, i.e. once FASTQ chunks, e.g. with a file size of 256MB, are generated by the NGS device. As a result, the data processing already starts while the sequencing run is still in progress. The results of the variant calling are stored in a task specific database table compatible to the Variant Calling Format (VCF) [18][19].

In the pipeline optimized for the IMDB technology the processing steps for sort, merge, and indexing are not performed by specific tools. These steps are directly executed by the IMDB without the need to create intermediate files in the file system.

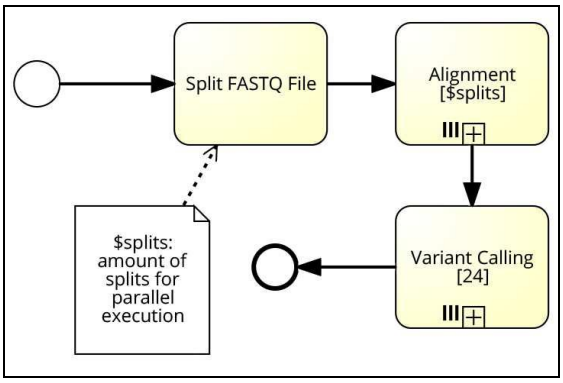

Fig. 6 Alignment algorithm which is called in parallel

IV. EVALUATION AND DISCUSSION

Fig 7 depicts the benchmark results comparing conventional data processing pipeline with various media breaks and the optimized pipeline incorporating the IMDB as integration platform. The throughput of the pipeline optimized for IMDB is about eight-times better than the throughput of the conventional pipeline. In contrast, the pipeline optimized for IMDB shows a constant scaling factor of approx. 1.04-1.10 for doubled input file sizes, i.e. the processing saturation for the benchmarked files was never reached. Furthermore, our benchmarks show that the IMDB optimized pipeline is able to process high-coverage FASTQ files in some minutes. For example, the largest input file with approx. 11kMbp was sequenced in approx. 45 minutes. For comparison, the conventional data processing pipeline took more than five hours to process the same file. All our benchmarks were executed on a cluster with 1,000 physical cores formed by 25 identical computing nodes. Each node is equipped with four Intel Xeon CPU E7-4870 Central Processing Units (CPUs) running at a clock speed of 2.40GHz providing a Quick Path Interconnect (QPI) speed of 6.4 GT/s. Each CPU is equipped 30MB of Intel's smart cache, ten cores, and 20 threads [20].

Parallel processing is much faster than sequential processing when it comes to doing repetitive calculations on vast amounts of data. This is because a parallel processor is capable of multithreading on a large scale, and can therefore simultaneously process several streams of data. Parallel processing is the simultaneous use of more than one CPU or

© 2014, IJERMT All Rights Reserved

processor core to execute a program or multiple computational threads. Ideally, parallel processing makes programs run faster because there are more engines running it. In practice, it is often difficult to divide a program in such a way that separate CPUs or cores can execute different portions without interfering with each other. Most computers have just one CPU, but some models have several, and multi-core processor chips are becoming the norm. There are even computers with thousands of CPUs.

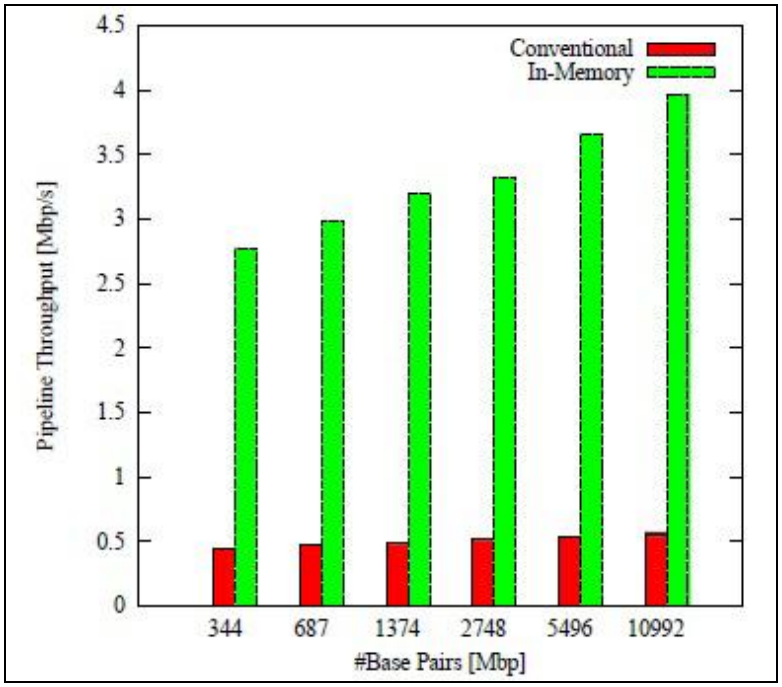

Fig. 7 Comparison of throughput of data processing pipeline consisting of whole genome alignment and variant calling.

V. CONCLUSION AND FUTURE WORK

Our work addressed various specific aspects of genome data processing. We showed that a tight integration of open-source tools for alignment and variant calling improves the overall throughput of the genome-processing pipeline. However, this integration requires a specific technology platform. We applied the IMDB technology as platform for integration of genome processing tools. The Genome Browser is a cross-platform desktop program for interactively visualizing high-throughput data in the context of the genome. Important features include dynamic panning and zooming, keyword search and open interoperability through the framework. Users may bookmark locations on the genome with descriptive annotations and share these bookmarks with other users. A key aspect of the Genome Browser is interoperability. To this flexible environment for exploring and combining data, the Genome Browser adds the ability to visualize diverse types of data in relation to its coordinates on the genome. Large scale collaborative efforts, multidisciplinary and international teams, comprehensiveness, high throughput data production and analysis, computational intensity, high standards for data quality, rapid data release, and attention to societal implications. It is clear that, the perfusion of genomics into other areas of biomedical research will enable these disciplines to make advances far beyond what is possible today. Building on our long lasting experience in applying in memory technology to selected enterprise challenges, we also focus on processing and analyzing of scientific data sets in real time. In particular, the applicability of in memory technology for analysis of genome data will be evaluated.

Future works will further improve the throughput of the processing pipeline by integrating tools into the IMDB technology, e.g. alignment or prediction mutation effects. As a result, we expect that the overall throughput of the genome data processing pipeline will improve further with these adoptions. There are several interesting directions for future work, we have seen that the bio informatics area can benefit a lot from conceptual modeling. But more work needs to be done to successfully propagate and apply the principles of good data design and physical data independence needs to the scientific community..

REFERENCES

[1] M. D'Antonio et al., "Building an Optimized Pipeline for Whole-exome Sequencing," EMBnet.journal, vol. 18, pp. 20–21, 2012.

[2] D. Crockford, "RFC4627: The Application/JSON Media Type for JavaScript Object Notation (JSON)," http://www.ietf.org/rfc/rfc4627.txt1, July 2006.

[3] M.-P. Schapranow et al., "Mobile Real-time Analysis of Patient Data for Advanced Decision Support in Personalized Medicine," in Proceedings of the 5th Int'l Conference on eHealth, Telemedicine, and Social Medicine (to appear), 2013.

[4] W. J. Ansorge, "Next-generation DNA Sequencing Techniques," New Biotechnology, vol. 25, no. 4, pp. 195–203, Apr. 2009.

© 2014, IJERMT All Rights Reserved

[5] K. Jain, Textbook of Personalized Medicine. Springer, 2009.
[6] M.-P. Schapranow, In-Memory Technology Enables History-Based Access Control for RFID-Aided Supply Chains. Springer London, 2013, ch. 9, pp. 187–213.
[7] A. Knopfel, B. Grone, and P. Tabeling, Fundamental Modeling Concepts: Effective Communication of IT Systems. JohnWiley \& Sons, 2006.
[8] R. W. W. Brouwer et al., "NARWHAL: A Primary Analysis Pipeline for NGS data," Bioinformatics, vol. 28, no. 2, pp. 284– 285, 2012.
[9] M.-P. Schapranow, C. Meinel, and H. Plattner, "Blitzschnelle Datenanalysen für die personalisierte Medizin," Laborwelt, vol. 13, no. 4, pp. 36–37, 2012.
[10] M. Z. DeMaere et al., "Simple High-throughput Annotation Pipeline (SHAP)," Bioinformatics, vol. 27, no. 17, pp. 2431– 2432, 2011.
[11] The Genome Reference Consortium, "Genome Assemblies," \\http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/data.shtml1.
[12] N. C. for Biotechnology Information, "All resources,"\\ http://www.ncbi.nlm.nih.gov/guide/all/1.
[13] F. S. A. et al., "Cosmic (the catalogue of somatic mutations in cancer): a resource to investigate acquired mutations in human cancer." Nucleic Acids Research, vol. 38, 2010.
[14] M. L. R. et al., "The UCSC Genome Browser Database: Extensions and Updates 2013," Nucleic Acids Res, 2012.
[15] A. T. Holdener, AJAX: The Definitive Guide, 1st ed. O'Reilly, 2008.
[16] Z. Su et al., "Next-generation Sequencing and its Applications in Molecular Diagnostics," Expert Review of Molecular Diagnostics, vol. 11, no. 3, pp. 333–343, Apr. 2011.
[17] S. S. T. et al., "dbSNP: The NCBI database of Genetic Variation, Nucleic Acids Res, vol. 29, no. 308–311, 2001.
[18] T. C. for Applied Genomics, "Database of genomic variants,\\ http://dgvbeta.tcag.ca/dgv/app/downloads1.
[19] M. Owen and J. Raj, "BPMN and Business Process Management,\\ http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf1, 2003.
[20] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in OSDI, 2004, pp. 137–150.

© 2014, IJERMT All Rights Reserved