Low latencies as conditione sine qua non for interactive data exploration and timely collaboration

Hajo N. Krabbenhöft[1,*], Steffen Möller[2]

[1]University of Lübeck, Institute for Neuro- and Bioininformatics, Lübeck, Germany

[2]University of Lübeck, Department of Dermatology, Lübeck, Germany

ABSTRACT

Motivation: High-throughput technologies, like gene expression arrays and next-generation sequencing, provide enormous data sets, which are too large to transfer or download quickly. The study of such data, for our application this means explaining the measurements with a molecular interpretation of disease etiology, requires continuous updates and refinements as novel interpretations are pursued. The complexity of the problem requires a diverse range of expertise. And thus a shared view is crucial for a successful collaboration - within and between institutions.

Web services and traditional web pages provide centralized data storage and synchronized presentation. Relying on a single central server, though, comes with its own flavor of reliability and performance issues. Every time the server is busy solving a request, the user is forced to wait. It is therefore very beneficial to combine the integrity of web services and the share-ability of web pages with the fluency of a desktop application. Increasing the interactivity of data presentation to each individual user allows for a more interactive knowledge exchange on the group scale.

Here, we present a combination of Open Source technologies for distributed, synchronized and failure-resistant storage of huge data sets as the technological basement for globally fast access to research data. Accordingly, this work explores the derived possibilities for interactive presentation to a group of locally distributed researchers, as enabled by a problem-tailored web application. To aid in the investigative work, the user interface shows minimal latencies. These goals are achieved by capitalizing on related developments in distributed data storage and asynchronous web technologies, most notably the non-relational database Apache Cassandra and the Google Web Toolkit. This combines efficient pre-processing with parallelisation.

The developed web application looks akin to a typical desktop application and is highly responsive, since it downloads needed data in parallel, while the user is happily working. The researcher can prepare different data set views for different aspects of his analysis, which are immediately available for colleagues and collaborators. By underpinning every decision and conference call with a synchronized shared data set, group communication is greatly improved. This work demonstrates that the interactivity with the user to work on large data sets is strengthened with remote applications and typical "show next page" delays are overcome by employing the latest web technologies. This way, the strong server-user interaction allows for the seamless extension for serving additional users and thus allow for collaborations.

Availability: Source code for the web application and the data storage back-end was released under GNU Lesser General Public License and is freely available for download from http://github.com/fxtentacle/eQTL-GWT-Cassandra

1 INTRODUCTION

The problem is as old as Bioinformatics itself: biological research yields more data than a human can handle manually.

Technological advancements in biochemistry and better insights in computational biology together have accelerated and broadened the avalanche of information. We are already losing this fight, as much data is generated in labs, which is never rendered available for analyses in other contexts, since the information is not publicly available. Given the inter-individual differences of patients and controls, many new model organisms, many more tissues investigated at an ever more detailed level and additional test conditions, the avalanche of usable knowledge will not stop in any foreseeable future.

For the analysis of large data, one needs to find human-digestible aggregations of it. It shall be a goal-oriented presentation where the essential information to form hypotheses is brought together. Further statistical evidence in the data will guide the downstream analysis, as will additional external information. For our application, the molecular interpretation of disease phenotypes, this is a process of continuous refinements and updates, accompanied by fruitful discussions with fellow researchers and collaborators. Every researcher needs software to look at and evaluate all possible measurements and explanations, without being forced to manually download or update the huge underlying data set. The

*To whom correspondence should be addressed.

researcher is doing investigative work and, therefore, it should be possible to browse all of the data interactively.

1.1 Prior art

The most common approach found in the Bioinformatics community for presenting and browsing data sets is to store research data inside a relational database and write custom-made software or web pages to present the data. Some approaches also include the ability to produce diagrams, but most are limited to text and tabular data. While generated web pages are usually static, this form of presentation is to be considered interactive if the user can issue filtering requests to the web server and retrieve a new web page containing the response within an acceptable response time.

Inside a relational database, data are stored as sorted by the primary key and separate look-up tables are generated for column range and equals queries. Since these look-up tables need to be identical for every database server in the system, common database software does not allow write requests to be distributed. There is, however, a range of solutions for replicating databases to multiple servers such that read requests can be distributed. This means that a relational database scales well with huge data sets when it comes to read requests and does not scale at all with write requests.

For very basic applications where research data are imported only once and then never modified, this works well. However, as soon as data are being processed or annotated on the user's behalf, the write rate of a single server is sustained. With traditional databases, only reads can be executed on a replica of the data and, therefore, the system as a whole cannot scale for writes.

We would like to see a word-processor like working environment: every data editable and inspectable with local views on the full data, helped by search tools and situation-dependent statistics like word counts. We want to avoid data transfer (takes too long), web-forms based "paged" interactions, any non-scalable components and want to directly support inter-user communication.

2 METHODS

The system evolved in Java from the PHP implemented TiQS interacting QTL System (tiqs.it). It does not require any local installation work apart from unzipping and executing a shell script. Web service requests are handled by a custom-written Java servlet inside a Jetty 6 (codehaus Foundation) servlet container. Jetty was chosen for its capability to run with the same configuration file on Windows, Mac OS X and Linux. Its direct competitor, Apache Tomcat, instead needs to be set up individually for each machine.

Data describing the topology of the system are stored in a PostgreSQL database (www.postgresql.org) with Hibernate 3 (www.hibernate.org). The valuable scientific data is stored inside a distributed Apache Cassandra (Lakshman and Malik, 2010) database. Load distribution is handled by a nginx load balancer (wiki.nginx.org), which was chosen for its high performance, stability and ease of configuration. Optimized example data set refinements, for the example of our expression QTL are provided. There is a plug-in API which allows researchers to write arbitrary filters and data processors in Java.

3 RESULTS

3.1 Minimal technical requirements

From the previous PHP implementation, no code could be saved and a separate execution environment was designed from scratch. It was taken special care to ensure the application to remain compatible with common IT constraints in research institutions: the application needs HTTP access on a random port for each worker node, as well as two configurable ports on which the peer-to-peer communication of the distributed database will take place.

The system can work with almost any memory and hard disk configuration and every Windows, Mac OS X and Linux computer can be turned into a worker node simply by copying a folder and running a shell script. The worker node software could also be deployed remotely.

The used database is data center aware in its distribution of redundancy. Accordingly, a complete self-contained copy of the data is kept at each physical facility, if the researcher has appointed one or more machines to use for data storage. The researcher can immediately start working, even while a local copy of the data set is being synchronized with coworkers and collaborators round the globe automatically. There is no initial waiting time for downloading the complete data set or manually deploying updates. Current Linux distributions like e.g. Debian (Möller et al., 2010) have all packages readily available or downloadable at the developers' websites.

3.2 Reliability and performance

Most current research software does not include any sort of failure tolerance and data replication, which comes as a surprise given the price of good research data. With growing data sets, storing and retrieving the correct subset is not a simple task anymore. A chain is only as strong as its weakest limb and therefore a bad database and schema choice will completely ruin a data exploration software.

Apache Cassandra was chosen because it is stable, fast and replicates data. This database is used in production at Facebook with billions of rows and therefore can be assumed to be reliable. Moving away from conventional relational databases towards a novel distributed system out of a key-value store and manually managed indexes payed off well.

Database query times average at 25ms and the system has shown a maximum in throughput of 10,000 expression QTL entries written per second on a single machine. Evaluating the performance on three machines showed that scalability was achieved and is simple to set up. When the database system is using the same configuration file on every machine, different worker nodes will automatically find each other and relocate the distributed data accordingly. Fault tolerance was evaluated by randomly disconnecting one of the three machines from the network. While the processing time for background tasks did go up when disconnecting an active worker node, the presentation front-end still responded as fast as before.

It was also verified that the data set stays complete and consistent as long as not more than half of the worker nodes are disconnected

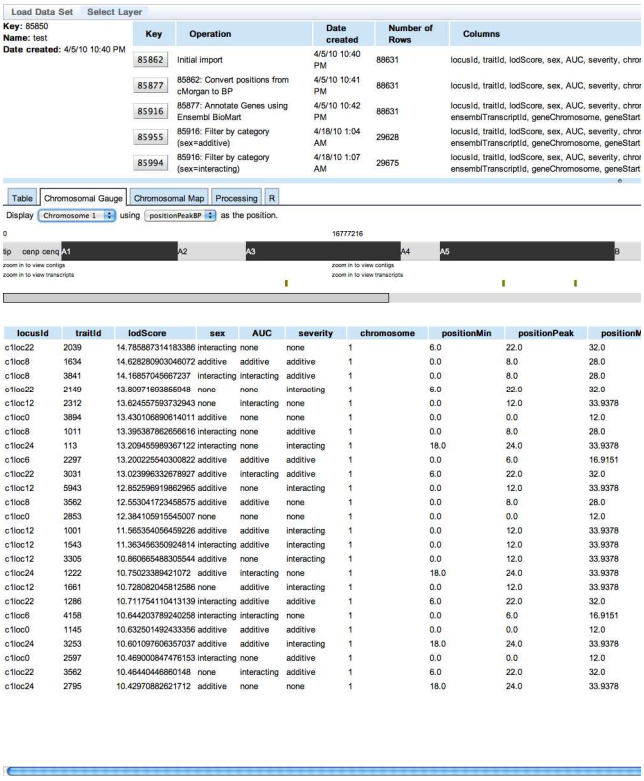

Fig. 1. The web application after loading a data set and selecting a data set layer. The chromosomal gauge view shows an overview of the chromosome, with important expression QTL locations marked by green/yellow blocks. Right alongside the overview, a table is displayed which shows the 25 most important expression QTL inside the area shown by the chromosomal gauge.

Fig. 2. The chromosomal map view. One can easily see which loci are interacting with which genes in the map area (F) as well as their border distribution (E). Note that the researcher has hidden the data set layer selector on the top to give more room for viewing the data. Shown are also the menu bar (A), the tab area for selecting a preferred presentation type (B), the settings for the chromosomal map presentation (C) and the chromosome bands, retrieved from Ensembl using DAS (D). On this zoom level, the scrollbars (G) are grayed out and the contig and transcript DAS tracks are inactive. The user can move click any displayed expression QTL or DAS track to get more information. The user can also drag and move the chromosomal map view and zoom in or out using the mouse wheel. This allows the user to dynamically switch between a genome-wide overview and a local detail view depending on the task at hand.

at the same time. So from a scalability and reliability point of view, this novel approach is superior to any conventional system relying on a single centralized database.

3.3 Example user interface for expression QTL

Above described technologies where applied for interactively presenting high-throughput data in statistical genetics. A web application provides a front-end to expression QTL data in a genomic context provided by DAS (Dowell et al., 2001). This ensures that users can easily share data with each other by sharing their links. By integrating a menu bar and by allowing the web page to be viewed in full screen, the user can interact with the web application akin to a desktop program. Since the whole program is run in the user's web browser, the user can use any operating system and does not require any prerequisites, except for the aforementioned web browser.

Figure 1 shows the developed web application running inside the web browser Google Chrome on Mac OS X. On the top, one can see the so-called data set layers. When the user invokes a filter or a processing operation, a modified copy of the shown data set is created and prepared for viewing in the background. The researcher can thereby prepare different presentations of his data for different aspects of his analysis. Since data set layers need to be computed only once, as opposed to workflows, for example, these views of the data set are immediately available for colleagues and collaborators.

On the lower half of the screen, one can see the chromosome browser with DAS tracks and annotations for the provided research data. One feature of the novel approach that was received especially well was the ability to change the viewing area in the chromosome browser without reloading the page. The user can click on and drag the chromosome to scroll. While the user is moving the display area using his mouse, the web application downloads the needed data in parallel, so it can update the view while the user is still scrolling. The table view below is also dynamically updated to

always show the most relevant data rows, in this case the 25 most probable expression QTL in the specific area of the chromosome,

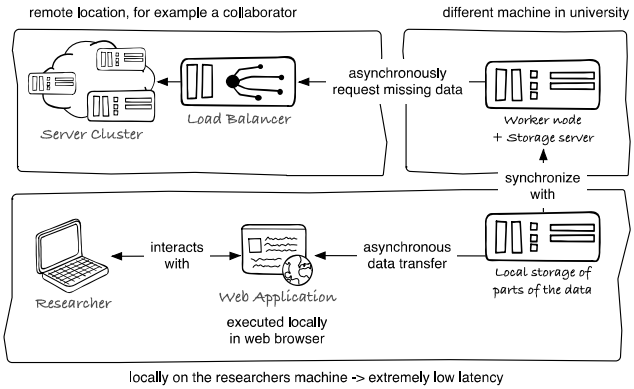

Fig. 3. Low latency is achieved by employing a traditional application model written in JavaScript to execute inside the users web browser. All data is being transferred asynchronously with a datacenter-aware multi-layered caching and replication strategy.

which is currently visible in the chromosome browser. The chromosomal map view shown in figure 2 was found to work great for getting a quick overview of which gene is interacting with which loci and later on investigating those locations. To ease the process of following up on these locations, transcripts and known genes retrieved from Ensembl DAS tracks are displayed alongside the expression QTL data when the user zooms in. The columns used for positioning along the X and Y axis can be freely chosen to accommodate different flavors of two-dimensional data.

The web application approach is highly reactive in comparison to normal web pages. That is presumably because most calculations are done when data are written. Hence, the data are stored already prepared, sorted and preformatted, which makes their presentation cheap in terms of network and CPU usage.

By working with a synchronized local copy of the database, read latency is as low as for traditional desktop applications. Each separate user benefits from this increased interactivity, thereby accelerating overall team communication.

3.4 How to obtain reliability, scalability and interactivity

3.4.1. Homogeneous replication of software and data

With exceptions for mobile computing, a local copy is always faster to access than relying on a remote service. However, users are not willing to wait hours or even days for a slow initial download of the whole data set. Therefore, our software system replicates commonly-accessed data automatically while falling back to remote access while replication or synchronization is in progress. This allows new user to immediately start working and by dynamically creating a local replica of the data set, we ensure that no row needs to be sent twice.

A research collaboration includes a number of computers distributed over several networks. It is safe to assume that on such a scale, at least one machine or network connection will fail. When scaling to work with huge data sets, even more computational power is needed and with more machines, component failures will get more and more often. In fact, we planned for and accepted them as part of the normal operation of a distributed software system.

For the system to stay operational and interactive under such conditions, the data need to be replicated. One simply cannot afford to lose data. Also, no worker node in the system should pose a single point of failure. Therefore, all nodes are running the same software and communicate with each other as equal peers. This approach is a stark contrast to the commonly used pattern of master-slave database replication. In cloud environments, this homogeneous configuration enables us to provide demand driven load balancing.

3.4.2. Copy on write

Not duplicating read-only data has been common sense in operating system design for decades, however with the growing amount of data stored in research databases, it is becoming increasingly important for maintaining a high read throughput. While creating a newly aggregated data presentation should preferably be fast, it is a rare event when compared to inspecting the data through already existing presentation views.

A research database should therefore rather create a modified and filtered copy of the data rather than using complex WHERE clauses and JOINs. If JOINs are unavoidable, most database systems provide a VIEW capability which gives the developer a warm fuzzy feeling of having thought ahead. Sadly, most VIEWs are not materialized by default. Therefore, performance of a standard database VIEW is as bad as calling the underlying JOINs and WHERE clauses on every access to any row of the VIEW.

This might seem trivial to state, but if one knows beforehand, that a certain VIEW will only be modified sparingly, that VIEW should be manually materialized. (CREATE TABLE … SELECT …) This trades a one-time creation overhead in return for significantly increased read throughput on following queries.

Using these two optimization techniques, the computational work can be moved from the presentation towards storage of the data, which allows for parallelisation and distribution on a compute grid. Keeping data stored the way it's supposed to be presented also ensures that all researches in the collaboration, independent of their available computing power, can immediately access and work with all presentations of the data.

3.4.3. Distribute work

Research data sets may easily contain thousands of rows. While enriching, annotating or filtering the data set, these rows can be processed independently. By distributing one-time computational tasks, such as the creation of a new presentation of the data, to all machines in the collaboration, everyone can see the result data set faster.

In our case, using a distributed database gives every worker node low-latency access on the whole data set and so one can actually

do workload distribution on a row scale. If you're forced to stick with a conventional relational database system, work distribution

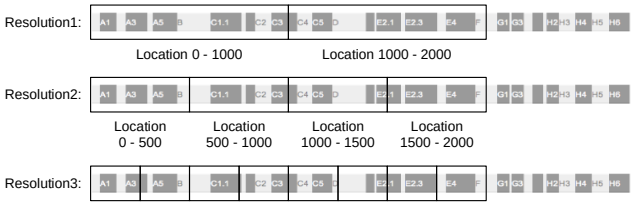

Fig. 4. Improving display performance by pre-calculating sorted lists of displayed items in order of decreasing relevance.

of low-level tasks might even be counter-productive or should at least be done with reasonably-sized blocks of data.

3.4.4. Prioritized non-blocking presentation as a stream of blocks of interest

Tables are the dominant form of presentation for research data. In Bioinformatics, gene locations and interactions play an important role and, therefore, different flavors of genome scales, chromosome browsers and interaction maps have been invented. Especially graphical presentations provide the researcher with a quick overview of his data. The possibility to zoom in and move around in a map of his data closes the gap between an overview of the complete data and a detailed close-up of specific features.

Interactive bars and maps constantly require new data to be shown, as the user is moving around and inspecting different aspects of the experiment. This puts an enormous strain on the back-end database, since it means a constant flow of search queries for aggregating the data to show and distance-comparisons are usually $O(N^2)$. It is also important, that only the most relevant information for a given zoom level is displayed, to make the resulting graphic not only sufficient, but also succinct.

While updating the data or creating a new presentation view, the location where each item will be visible on the bar or map is usually known beforehand. Similarly, when applying our suggestion of copying the data on write time, the relevance of every item for every zoom level can also be calculated offline.

We therefore propose to divide the possible view area into a hierarchical set of equally-sized blocks, as seen in figure 4. Assuming that display position and relevancy score have already been calculated, the data should look akin to table 1.

We now compare these positions to figure 3. On zoom resolution 1, all items are in the first block. The same applies to zoom resolution 2. On resolution 3, the item with ID=1 is in the second block, while the other two items are in the first block. Now we can create a look-up table for every block at every resolution, which is easy given the schema-less nature of our chosen distributed key-value store. Example look-up tables are shown in table 2. Please note that the rows of each table have been sorted by their relevancy scores.

Relevancy scores are stored alongside in the index table, to allow for dynamic merging of blocks. If the user requests the range 0-500

Table 1. Example data

ID	Position	Score	…
1	394	14	
2	112	3	
3	113	5	

Table 2. Example look-up tables

Resolution 1, Block 1	Score
1	14
3	5
2	3

Resolution 3, Block 1	Score
3	5
2	3

Resolution 3, Block 2	Score
1	14

on resolution 3, we could dynamically merge the tables "Resolution 3, Block 1" and "Resolution 3, Block 2". Given the hierarchical nature of our block scheme, that request would, of course, be easier satisfied by using "Resolution 2, Block 1".

Now, for dynamically filling the visual presentation, items are streamed for the requested zoom level. The streaming follows the order of those tables, presenting the most relevant rows first and filling the image as remaining data arrives. The client application can then choose itself how many items it needs to adequately populate the view and close the connection when enough data was received. This allows or web application, for example, to dynamically adapt the number of displayed expression QTLs to the users display resolution.

4 DISCUSSION

Setting aside the parallel database, the major contribution towards a new sense of responsiveness is due to the selective transfer of information of blocks from the server to the user. This is what a local application would also attempt to perform, but what traditional web forms just cannot achieve when they rebuild the page from scratch.

This way, the introduction of JavaScript - well hidden behind Java classes by the Google Web Toolkit - contributed far more than just the usual eye candy. The approach was so much more responsive than traditional PHP-produced tables, showing the same information, that we have not even taken measurements. It was "instant" versus "wait a few seconds". Also, the approach was found to consume less bandwidth. We hence expect this sort of web applications to be adopted by many public Bioinformatics databases throughout the next years.

The technologies described above are used in a series of well known Internet sites like Facebook or the Google family of web-based applications. With these prime examples in mind, and tapping into our experiences gained over this implementation, we

shall compare other contemporary eQTL infrastructures with what they could achieve, when they adopted those technologies.

4.1 XGAP

A related project is the eXtensible Genotype And Phenotype platform (XGAP) (Swertz et al., 2010). The XGAP project aims to provide a flexible and open platform for working with data sets, specifically developed with expression QTL data in mind. The XGAP project aims to make working collaboratively easy and provides integrated tools for importing and exporting data.

It is noteworthy, that the XGAP project shares many design decisions with the web application approach presented here. For example, developing a web page rather than a program gives researchers the freedom to share the interface and therefore access to the data without requiring the recipient to have a matching operating system and sufficient processing power available. It is expected, that more and more applications and data interfaces in general will be developed as web pages to shift the burden of installation and configuration from the users towards the software developers.

The second shared approach is that of grid computing and background processing. XGAP supports invocation of computationally intensive tasks asynchronously in the background, with the workload distributed on a PBS cluster. The novel approach presented in this paper incorporates asynchronous distributed processing as a core feature and can therefore support load-balancing and failure tolerance at a level deeper than XGAP. It is expected that this trend will continue and eventually flash over to consumer applications. This is essential in order to reap the full benefits of newer processors, which come with more and more cores whereas further increasing the frequency is getting more and more difficult.

XGAP completely lacks a distributed data storage. When their MySQL database as a storage back-end breaks down, the XGAP system will loose all of its data, therefore presenting a single point of failure. Neither is it prepared for parallel data management.

4.2 Gene Network

Another related project is Gene Network (Wang et al., 2003). Gene Network providing follow-up information about genes, loci and gene networks and their module WebQTL allows the user to upload own research data for further analysis. Gene Network says to archive more than 25 years of research data and provides a very good coverage of additional information.

Obvious shortcomings of the Gene Network are that data transfer is not being encrypted using the industry standard HTTPS and that there exists no version which the researchers could deploy on-site inside their own firewall. Apart from security issues, WebQTL allows for a very convenient analysis of small data sets. It provides a plentiful selection of visualization methods, such as box plots, correlation diagrams and even directed graphs.

From a technical point of view, the Gene Network is considered to be inferior to both the novel approach presented as well as XGAP. Data set presentation in Gene Network is implemented as downloading ready-made images from their web servers. This leaves the user with no further possibility for interaction than changing parameters and waiting for the next image to be downloaded. Since the researcher has no possibility of running Gene Network on own computational resources, the web servers provided by Gene Network are essentially shared by all users.

In its choice of presentation methods, Gene Network is very similar to the R language for scientific computing. Due to Gene Networks focus on reasonably small data sets, using such a scripting language as computational back-end seems a wise choice. Gene Network is there wholeheartedly recommended for analytical and investigative work on classical QTL.

4.3 Extendability of presented concepts

By going new ways in terms of data storage, we combined the low latency of local data storage with the benefits and integrity of a centralized storage server. This technological design decision allowed us to greatly increase interactivity of our data presentation, without forcing the user to download the complete data set beforehand. While during development, there was a strong focus on expression QTL, that is their positioning on the chromosome and their associated genes, the system was designed to be plug-able for a multitude of data processors and data visualisation applications.

The chromosome browser allows for arbitrary chromosomes to be displayed along with arbitrary annotation information, as long as the DAS file format is being used. Similarly, the map view allows for arbitrary positioning measures to be used on the X and Y axis, as long as there is a data processor available to calculate said positions. While theoretically any user could develop such data processors using a simple Java API, it might be beneficial to broaden our showcase of example processors to support additional forms of high-throughput data.

Data replication allows a whole team a consistent shared view of their experiment. A new presentation created by one collaborator is immediately available to the entire team. Driven by the high interactivity between every user and the web application, overall team communication is speeding up, and there is a general demand for a deeper integration of social aspects into the data presentation. We envision a future version of our system where researchers can discuss current and past measurements in real-time using a special comment and annotation system adapted to work directly on the data presentation.

With computing nodes gradually getting cheaper and more readily available, dynamic grid brokering will replace static worker queues and present us with unprecedented peak amounts of compute power. While grid technologies traditionally suffer from their own transiency, the distributed and homogeneous nature of our proposed system can easily compensate for node failures, while still retaining near-perfect performance.

ACKNOWLEDGEMENTS

The authors thank Thomas Martinetz and Saleh Ibrahim for comments and a nice working atmosphere. Lydia Lutter is thanked for her critical reading of the manuscript.

REFERENCES

codehaus Foundation. Jetty 6 http server. URL http://jetty.codehaus.org/jetty/.

William Cookson, Liming Liang, Goncalo Abecasis, Miriam Moffatt, and Mark Lathrop. Mapping complex disease traits with global gene expression. Nat Rev Genet, 10(3): 184–194, 03 2009.

Robin Dowell, Rodney Jokerst, Allen Day, Sean Eddy, and Lincoln Stein. The distributed annotation system. BMC Bioinformatics, 2(1):7, 2001. ISSN 1471-2105. doi: 10.1186/ 1471-2105-2-7

David Flanagan. JavaScript: The Definitive Guide. O'Reilly Media, Inc., 2006.

Ewald Geschwinde and Hans-Jürgen Schonig. Postgresql Developer's Handbook.

Sams, Indianapolis, IN, USA, 2001.

International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. Nature, 431(7011):931–945, Oct 2004.

FA Kolpakov, EA Ananko, GB Kolesov, and NA Kolchanov. GeneNet: a gene network database and its automated visualization. Bioinformatics, 14(6):529–537, 1998.

Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. SIGOPS Oper. Syst. Rev., 44(2):35–40, 2010.

Steffen Möller, Hajo Nils Krabbenhöft, Andreas Tille, David Paleino, Alan Williams, Katy Wolstencroft, Carole Goble, Richard Holland, Dominique Belhachemi, Charles Plessy. Community-driven computational biology with Debian Linux. BMC Bioinformatics 11 Suppl 12 (2010): S5.

M Rosenberg and D Court. Regulatory sequences involved in the promotion and termination of rna transcription. Annu. Rev. Genet, 13:319–53, 1979

R Sachidanandam, D Weissman, S C Schmidt, J M Kakol, L D Stein, G Marth, S Sherry, J C Mullikin, B J Mortimore, D L Willey, S E Hunt, C G Cole, P C Coggill, C M Rice, Z Ning, J Rogers, D R Bentley, P Y Kwok, E R Mardis, R T Yeh, B Schultz, L Cook, R Davenport, M Dante, L Fulton, L Hillier, R H Waterston, J D McPherson, B Gilman, S Schaffner, W J Van Etten, D Reich, J Higgins, M J Daly, B Blumenstiel, J Baldwin, N Stange-Thomann, M C Zody, L Linton, E S Lander, D Altshuler, and International SNP Map Working Group. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. Nature, 409(6822):928–33, Feb 2001. doi: 10.1038/35057149.

Jay Shendure, Gregory J. Porreca, Nikos B. Reppas, Xiaoxia Lin, John P. McCutcheon, Abraham M. Rosenbaum, Michael D. Wang, Kun Zhang, Robi D. Mitra, and George M. Church. Accurate Multiplex Polony Sequencing of an Evolved Bacterial Genome. Science, 309(5741):1728–1732, 2005.

Morris Swertz, K Joeri Velde, Bruno Tesson, Richard Scheltema, Danny Arends, Gonzalo Vera, Rudi Alberts, Martijn Dijkstra, Paul Schofield, Klaus Schughart, John Hancock, Damian Smedley, Katy Wolstencroft, Carole Goble, Engbert de Brock, Andrew Jones, and Helen ... Parkinson. Xgap: a uniform and extensible data model and software platform for genotype and phenotype experiments. Genome Biology, 11(3):R27, 2010.

J M Trent, M Bittner, J Zhang, R Wiltshire, M Ray, Y Su, E Gracia, P Meltzer, J De Risi, L Penland, and P Brown. Use of microgenomic technology for analysis of alterations in dna copy number and gene expression in malignant melanoma. Clin. Exp. Immunol., 107 Suppl 1:33–40, Jan 1997.

Jintao Wang, Robert W Williams, and Kenneth F Manly. Webqtl: web-based complex trait analysis. Neuroinformatics, 1(4):299–308, 2003.

W3C Consortium. Soap version 1.2 part 1: Messaging framework (second edition), a. W3C Consortium. Web services description language (wsdl), b.

Adam Warski. Envers: Easy entity auditing. URL http://jboss.org/envers/. 37

Michael Widenius, David Axmark, and A. B. Mysql. MySQL Reference Manual. O'Reilly Media, Inc., 1st edition, 2002.

Technical diagrams have been created using the "Architecture by Hand" stencil set by Jonathan Brown.