AJAX Interface: A Breakthrough in Bioinformatics Web Applications

G. Aravindhan[1], G. Ramesh Kumar[1], R. Sathish Kumar[2] and K. Subha[1]

[1]Bioinformatics Division, AU-KBC Research Centre, MIT Campus, Anna University, Chennai-600 044, India. [2]NRCFOSS, AU-KBC Research Centre, MIT Campus, Anna University, Chennai-600 044, India.

Abstract: Bioinformatics applications are generally multi-server dependants and will have to communicate several information repositories to carry out any analyses. These applications remain computationally intensive and time consuming as they engage lots of data transfer. Hence they face a major bottleneck when ported as web applications. Browser based web applications normally feature the classical request-response approach. If the response becomes late, as it is expected to happen in the case of long running Bioinformatics programs, Apache will get tired and a request timeout error might occur. Alternate approaches like "Client-Pull" models that involve polling strategy with the unpredictable amount of page refreshes, only tend to intensify the network traffic. Hence a technology that is intelligent enough to support the varied exhaustive Bioinformatics processes becomes highly essential. In this review, we propose how AJAX can afford a laconic framework within the Bioinformatics applications to completely reduce the page refresh nuisance and provide a better user experience.

Introduction

Vast amounts of biological data have been generated in the last decade[1] and now the research focus is towards analyzing and deriving some meaningful information from these hugely accumulated raw data. Such analyses and interpretation remain the sphere of influence of the burgeoning field, Bioinformatics. The major objective of Bioinformatics is to extract useful knowledge from the flood of data, including biological literature, for the purpose of further analysis ultimately leading to drug discovery and other useful applications. Bioinformatics generally involves a variety of resources such as tools and databases, operating on the diverse algorithms, to explore the Bio-information. There is a great proliferation of Bioinformatics resources that allows the users to browse, search and analyze the data they contain.[2] Most of the Bioinformatics applications prevailing today are only the interfaces between the data repositories and the client systems by means of web pages featuring a lot of information transfers. These factors make the Bioinformatics applications more computationally intensive and time consuming. Hence they face major difficulties when ported as web applications. Web applications used in Bioinformatics often has different requirements, since they quite frequently has a very high computational load per access, and an access may require long lasting computations before the result can be presented to the user. Also, the application core is frequently more complicated than what is common for web applications. For example, a simple similarity search or a domain search of a single protein sequence will itself take several minutes to produce the result. Until, the result of the query sequence is ready, the browser keeps on refreshing repeatedly, which remains unpleasing and also time consuming.

Classical Approaches

The classical web application model follows the request response cycle of HTTP, Hyper Text Transmission Protocol.[3] This request-response paradigm follows a three-step process. As a first step, the user sends a HTTP request to the server and when a user's browser is directed to an URL, it connects to the server mentioned and sends it with a GET request that includes the path to the required filename. In some cases, the request may also contain POSTed data. In the second step, the server carryout the processing of the query based on the algorithm and finally the results are returned to the Client browser as HTML pages.[4] Though this approach makes a lot of technical sense, it does not provide a great user experience. This is

Correspondence: G. Aravindhan, Centre for Molecular Simulations, Faculty of ICT, Swinburne University of Technology, Victoria-3122, Australia. Email: aganesan@groupwise.swin.edu.au

because, when a browser sends a request to the server, it keeps an open connection for receiving the response from the server. In such cases, if the server does not send any response for a stipulated time, the browser will time-out without collecting any reply from the server and results in the timeout error or 408 Request Timeout error. Hence this approach will not be suitable for the heavily time consuming Bioinformatics applications.

Polling with Page Refresh

A common accepted approach to resolve the time out error issue is polling the server with page refresh [http://www.freepatentsonline.com/7330894.html]. This solution that minimizes server overhead and client browser dependence is the use of "client pull," also called "Meta refresh''. The initial request sets up a forked process to perform the long running process, and redirects the browser to a status page. The status page refreshes itself every n seconds through the meta-refresh header (<meta http-equiv = "Refresh" content = "15">). Each time it refreshes, it polls the status of the process at the server. If the process is not complete, the browser is instructed to "refresh.''

If the task is complete, the browser is directed to the results page without the page refresh attribute. Though an elegant solution, its major drawback is the superfluous network traffic, as each refresh involves a completely new request to a server. Every active client sends a request to the server every n-second. Even a status page, featuring request/response will contain many hundreds of bytes. Therefore, for an example, if 1,000 active users are polling every second at a 200-byte payload, then we would be using 200KB/s (1.2 mbps–almost 500 GB per month) of bandwidth. Moreover the periodic page refresh isn't visually appealing either. As most of the Bioinformatics applications are generally multi-server dependants, they will have to exchange data with many information repositories and hence the complexity involved in these processes remains unavoidable.

Although there are several Bioinformatics programs that deploy page refreshes, it is worth evaluating BLAST,[23,28] a similarity search tool as it is the most frequently used Bioinformatics tool worldwide. Searching a single protein sequence against the NCBI BLAST utility, offered by the European Bioinformatics Institute, [http://www.ebi.ac.uk/Tools/blast/] will itself employ voluminous amount of data transfer. Once the query is submitted in the BLAST utility, the browser will submit it to the server where the similarity searching takes place. When the process is being carried out, the browser will display a window showing the page refresh (Fig. 1), which is not pleasant. As the page refresh is taking place, the user must sit idle and wait for the results to appear in the browser. Hence this process eats up the user's time and also the user's money as it consumes a lot of bandwidth. Hence a technology that is more compatible to support the complex Bioinformatics applications becomes the need of the hour.

AJAX-A novel technology for Web Applications

AJAX, the brain child of Jesse James Garrette of Adaptive Path, LLC was crafted and published in 2005 as "AJAX : A New Approach to Web Applications".[4] The acronym AJAX stands for Asynchronous Javascript and XML. AJAX is not a completely new technology but still the implementation of AJAX is novel. It is just the combination of several existing technologies such as HTML, CSS, DOM, XML, XSLT, XMLHttpRequest and Javascript. The AJAX interfaced applications eliminate the traditional request-response-request-response nature of web pages and hence allowing them to behave like the desktop applications until limited extent. AJAX allows pages to request small bits of information from the server instead of entire pages. In this new model, the web interface is made up of individual elements that can be updated or replaced independently and does not require the entire page to be reloaded on every user action.[5] The principle of AJAX is that instead of loading a web page, at the start of the operation, the browser loads an AJAX program, which is written in JavaScript and normally made hidden. This AJAX program serves as an interface between the User and the server and thereby allowing the user's interaction with the application.[4] The communication between client and server can take place asynchronously, which is substantially different from the classic synchronous request, wait for response, and continue model (Fig. 2).[5,6]

At the core of the AJAX paradigm is XMLHttpRequest, a Javascript object that permits transfer of XML[7] over HTTP[3,8,9] XMLHttpRequest can make asynchronous requests to the server. XMLHttpRequest's onreadystatechange event is

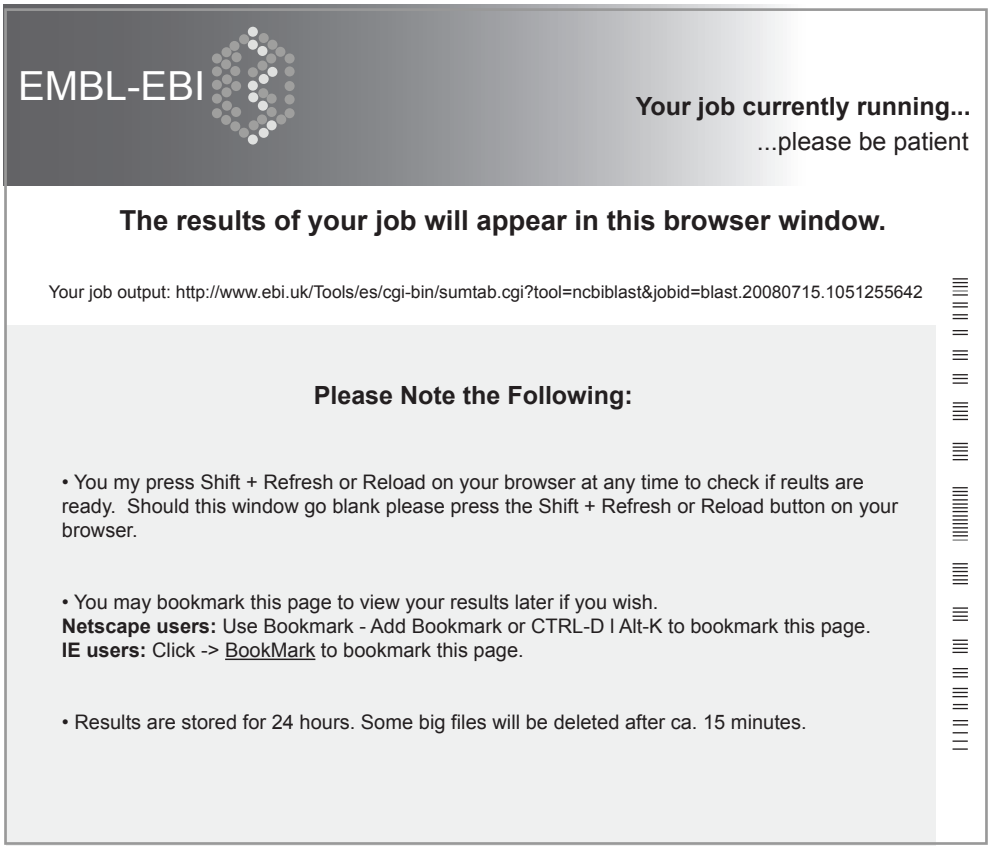

Figure 1. Page Refresh when using EBI-NCBI BLAST.

binded to a callback function that will monitor the status of the response. This will enable invoking appropriate code once the response is ready. The response is in the form of XML,[7] which is parsed, and the Document Object Model (DOM)[10] manipulated by Javascript[11] code to modify just a part of the web page without complete page reload. Thus AJAX enabled pages are without page refresh thereby providing rich desktop like experience to its users thereby preventing the users from sitting idle while the query process is taking place and, in turn, helps to increase the levels of interactivity, responsiveness and user satisfaction.[4,5,12]

Hence AJAX based techniques have become more serious option not only for newly developed applications, but also for existing web sites if they are not enough user-friendly. Google Products such as Gmail (http://mail.google.com/), Google Maps (http://maps.google.com), and Google Suggest [http://www.google.com/webhp? complete=1],[13] flickr (www.flickr.com), a free online image and video hosting website[14] and Weebly (www weebly com), a free online web creation suite[15] are all some of the popular and successful web applications that are interfaced with the AJAX technology.

AJAX Prototype in Bioinformatics

The need for Bioinformatics has arisen from the recent explosion of publicly available genomic information. Such an outburst in the biological information has forced the development of novel algorithms and software tools to analyze them. Until date, a variety of Bioinformatics tools and databases have been developed. They facilitate and quicken the analysis of the systems level processes.[16]

Major approaches that are used in Bioinformatics are (i) the use of computational searches and alignment techniques[17,18] to compare the unknown genes with the set of known genes, (ii) use of mathematical modeling techniques such as data mining and (iii) an integrated approach that combine various search techniques with the mathematical modeling. In all these approaches, the principle is that, the information source will be

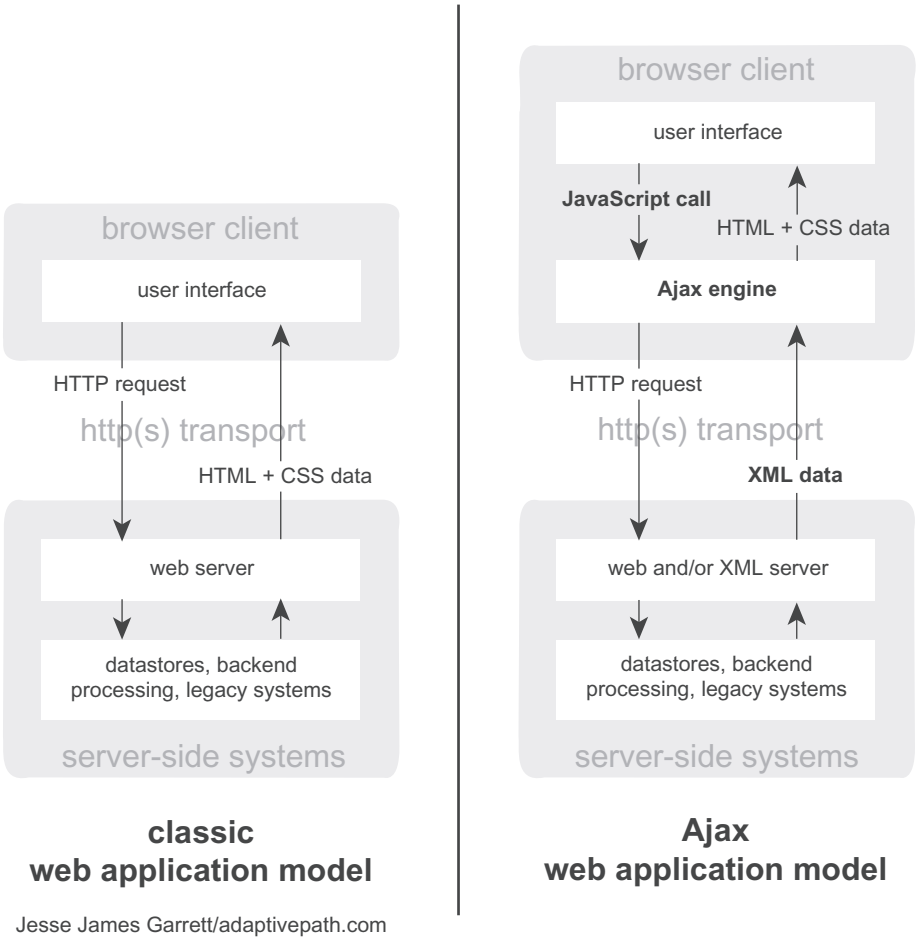

Figure 2. AJAX model vs. classical model.[6]

available in the form of different databases and with the use of variety of tools, operating on diverse algorithms, these known information will be utilized to analyze the unknown or anonymous genes. Such types of Bioinformatic methodologies utilize a simple workflow that typically accept an input file, analyze to compute a result and finally present an output file to the users.[19] Though the workflow appears to be simple, the Bioinformatics tools engage a lot of complicated algorithms that makes the analyses more intensive and face a lot of problems when hosted as web applications. Further, since the information repositories prevail to be the unique source for the entire Bioinformatics research community, a lot of network traffic occurs while utilizing the information servers.

For example, in any sequence analysis tools, when a query sequence is submitted, it generates a Job ID for the submitted sequences and then proceeds for the analysis. Then the browser will POST the HTTP Request to the server and maintains an open connection to receive the response from the server. At this intermediate period, the periodic server polling will take place through Meta-Refresh during which the users are forced to sit inactively and just stare at the refreshing window. Only after the results are ready at the server, the HTTP[3,8] will fetch the result and display it to the user. Sometimes, the browser will not receive any response for stipulated period and will time-out without a response resulting in the 408 Request Timeout errors. Such visually unappealing page refresh nuisance of long running bioinformatics web applications can be done away with using a new AJAX design pattern (Fig. 3). The uniqueness of AJAX technology is the paradigm shift from traditional web page-refresh model to Rich Internet Applications (RIA) mimicking desktop like instant response and User interface. Though the core philosophy of AJAX is RIA, AJAX has slowly found a novel niche in the Bioinformatics web applications.

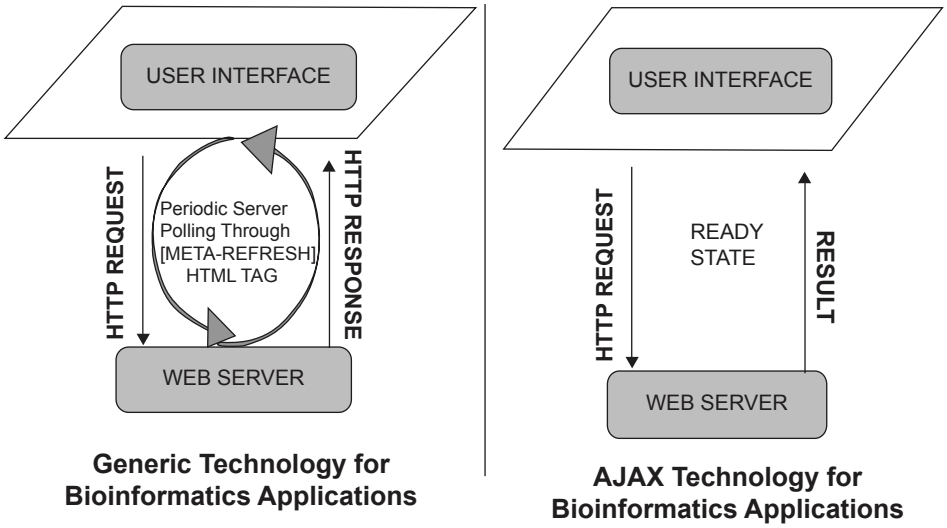

Figure 3. Generic technology versus AJAX technology for bioinformatics applications.

In this AJAX pattern, the XMLHttpRequest object binds to a callback Javascript[11] function and then sends a POST or GET request to the server asynchronously. The handler function monitors the readyState property of XMLHttpRequest that changes as the request goes through and the response is received. Until the readyState becomes 4 (meaning that the response has been completely received) a progress bar is displayed to signal the progress of the long running process. Once readyState is 4, the callback handler gleans the results out of the response XML and displays the result by DOM manipulation without page refresh.

Examples of Bioinformatics Systems with AJAX Technology

There are already a number of bioinformatics systems that have successfully implemented the AJAX technology. MutDB (http://www.mutdb.org), a tool aiming to aid bioinformatic studies by integrating publicly available databases of human genetic variation with molecular features and clinical phenotype data have utilized AJAX technology to execute a paging scheme to increase the responsiveness upon large data sets. Using AJAX, this database has successfully enhanced the speed of data exchange with the server.[20] Whereas, AVIS—AJAX viewer of interactive signaling networks [http://actin.pharm.mssm.edu/AVIS2], a google gadget compatible web-based viewer of interactive cell signaling networks have employed AJAX with the usage of the libraries GraphViz, ImageMagic (PerlMagic) and overLib to increase its capability of visualizing the text-based signaling networks in no time.[21] Bobby-Joe Breitkreutz and his group have developed the Biological General Repository for Interaction Datasets (BioGRID) database (http://www.thebiogrid.org) to house and distribute collections of protein and genetic interactions from major model organism species. They have powered BioGRID by AJAX to make the downloads for every search result page are available in the database.[22] Bioinfortracker [http://biotool.nrcfosshelpline.in/] is a co-ordinated program for performing functional annotation of the genes from any organism based on the different functional annotation strategies such as Blast,[23] Pfam,[24,29] InterproScan,[25] and COG.[26] This tool has effectively used AJAX as an interface and greatly reduced the consumption of Bandwidth. CARGO [http://cargo.bioinfo.cnio.es.], an advanced web portal to integrate the biological information is also designed with the support of a Rich Internet Application (RIA) paradigm based on AJAX.[27]

AIM-BLAST: An AJAX Interfaced Multisequence Blast

AIM-BLAST-an AJAX Interfaced Multisequence Blast [http://biotool.nrcfosshelpline.in/blast/] is another example for an advanced Bioinformatics system, which has explicitly interfaced AJAX to carryout multiple protein sequences search at an instance. To ensure that the efficiency of the AJAX based sequence similarity search is better than

Table 1. Comparison of total data transfer and the overall processing time between EBI-NCBI BLAST and AIM-BLAST for the sample set of sequences.

Tools	EBI-NCBI BLAST	AIM-BLAST
Data transfer during analysis		
Data sent [in Bytes]	818769	49488
Data received [in Bytes]	11800797	30596
Total data transfer [in Bytes]	12619566	80084
Total data transfer [in Mega Bytes]	12.619566	0.080084
Processing time during analysis		
Time [in Minutes]	96.32	12.03

the regular Blast search, the performances of AIM-BLAST and NCBI BLAST were compared simultaneously using a sample set of 30 protein sequences of varying length from *E.coli K12 strain*.

Both the AIM-BLAST and the EBI-NCBI BLAST were run in the Firefox Web browser and the HttpFox (https://addons.mozilla.or/en-US/firefox/addon/6647), a Firefox add-on was operated at the backend to measure the amount of bandwidth consumption during the analysis. As soon as the analysis of the entire set of sequences was completed, the loads of bytes sent and received for the sample protein sequences was tabulated (Table 1) for comparison. As per the resultant table, it was observed that EBI-NCBI Blast consumed an overall data transfer of 12.62 Mega Bytes viz. 0.8 MB of data sent to the server and 11.80 MB of data received from the server for analyzing just 30 sample sequences. Whereas, AIM-BLAST consumed only 0.08 Mega Bytes of data transfer viz. 0.049 MB of data sent to the server and 0.031 MB of data received from the server. Moreover, as in regular blast service, the extensive book keeping at the server to keep track of jobs and job-ids is not required for AIM-BLAST and this ensures that this tool reduces the superfluous network traffic and saves bandwidth. Thus, this real-time analysis ensured the advantages that AJAX interface could provide for any Bioinformatics web applications.

All these systems clearly show that executing AJAX as an interface in the bioinformatics applications will make the processing more convenient and impressive. Above all, implementation of AJAX with the web applications will greatly control the consumption of bandwidth. Further, the proposed approach has a number of merits compared to traditional model. Superfluous network traffic is reduced due to the absence of polling. The extensive book keeping at the server to keep track of jobs and job-ids is not needed here. The precarious and visually unappealing page refresh has been replaced by a simple progress bar that is a really effective user interface pradigm. The AJAX based approach has some demerits too. AJAX cannot maintain bookmarks or browser history. As the results are dynamically generated by manipuating Document Object Model (DOM), the results page cannot be saved in the normal way. Still these demerits specific to AJAX, have a number of proposed solutions. Hence, no doubt, AJAX will take up principal role in the Bioinformatics applications in future.

Conclusion

With a lot of sophisticated functionality using easier to implement web standards, AJAX will be a real alternative for creating more powerful and user-friendly Bioinformatics applications. With too many merits to offer, AJAX will make the Bioinformatics world quicker and more notable. Last but not least, we conclude that AJAXification of Bioinformatics applications will become inevitable.

Disclosure

The authors report no conflicts of interest.

References

1. Kim Carter and Matthew Bellgard. MASV—Multiple (BLAST) Annotation System Viewer. *Bioinformatics*. 2003;19(17): 2313–2315.
2. Kevin Garwood. et al. Model-driven user interfaces for bioinformatics data resources: regenerating the wheel as an alternative to reinventing it. *BMC Bioinformatics*. 2006;7:532.

3. William Stallings. Introduction to Hyper Text Transfer Protocol. *InformIT*. 2001.
4. Garrett J. AJAX: A new approach to web applications. Adaptive path, http://www.adaptivepath.com/publications/essays/archives/000385.php. 2005.
5. Ali Mesbah. AJAXifying Classic Web Applications, Companion to the proceedings of the 29th International Conference on Software. *Engineering*. 2007;81–82.
6. Mesbah A, van Deursen A. An architectural style for AJAX. In WICSA'07: 6th Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society. 2007.
7. Bray T, et al. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/1998/REC-xml-19980210. 1998.
8. Chisholm W, et al. Web Content Accessibility Guidelines 1.0. http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/. 1999.
9. Fielding R, et al. Hypertext Transfer Protocol—HTTP/1.1. The Internet Society. http://www.w3.org/Protocols/rfc2616/rfc2616.html. 1999.
10. Byrne et al. Document Object Model (DOM) Level 1 Specification http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001. 1998.
11. Lou Montulli montulli et al. JavaScript-Based Style Sheets. http://www.w3.org/Submission/1996/1/WD-jsss-960822. 1996.
12. Mesbah A, van Deursen A. Migrating multi-page web applications to single-page AJAX interfaces. In CSMR '07:11th European Conference on Software Maintenance and Reengineering. IEEE Computer Society. 2007.
13. Battelle, John. The Birth of Google. *Wired Magazine*. 2005.
14. Auchard. Eric Flickr to map the world's latest photo hotspots. *Reuters*. 2007.
15. Levy Stephen A. Boot Camp for the Next Tech Millionares. *Newsweek*. 2007.
16. Arvind K Bansal. Bioinformatics in microbial biotechnology—a mini review. *Microb Cell Fact*. 2005;4:19.
17. Pearson WR, et al. Improved tools for biological sequence comparison. *Proceedings National Academy of Science U S A*. 1988;85:2444–2448.
18. Waterman MS. Introduction to Computational Biology: Maps, Sequence, and Genomes. Chapman and Hall, London. 1995.
19. Alexander Garcia Castro, et al. Workflows in bioinformatics, meta-analysis and prototype implementation of a workflow generator. *BMC Bioinformatics*. 2005;6:87.
20. Arti Singh et al. MutDB:update on development of tools for the biochemical analysis of genetic variation. *Nucleic Acids Res*. 2008;36:D815–D819.
21. Seth I, Berger, et al. AVIS: AJAX viewer of interactive signaling networks. *Bioinformatics*. 2007;23(20):2803–2805.
22. Bobby-Joe Breitkreutz, et al. The BioGRID Interaction Database: 2008 update. *Nucleic Acids Res*. 2008;6:D637–D640.
23. Altschul SF, et al. Basic local alignment search tool. *J Mol Biol*. 1990;215(3):403–410.
24. Cathy H, Wu, et al. Protein family classification and functional annotation. *Comput Biol Chem*. 2003;27(1):37–47.
25. Mulder NJ, et al. New developments in the InterPro database. *Nucleic Acids Res*. 2007;35:D224–8.
26. Tatusov RL, et al. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*. 2003;11;4:41.
27. Cases Ildefonso, et al. CARGO: a web portal to integrate customized biological information. *Nucleic acids research*. 2007;35:W16–W20.
28. Altschul SF et al. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*. 1997;15:3389–3402.
29. Bateman A et al. The Pfam protein families database. *Nucleic Acids Res.* 2004;1:32 D138–141.