RECIIS
Electronic Journal of Communication Information & Innovation in Health
[www.reciis.cict.fiocruz.br]
ISSN 1981-6286

SUPPLEMENT – BIOINFORMATICS AND HEALTH

Original Articles

ASKGene, a system for automate DNA processing

DOI: 10.3395/reciis.v1i2.Sup.100en

Eden Cardim
Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas da Universidade Estadual de Santa Cruz, Bahia, Brazil
edencardim@gmail.com

Wallace Reis
Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas da Universidade Estadual de Santa Cruz, Bahia, Brazil
reis.wallace@gmail.com

Nicolas Carels
Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas da Universidade Estadual de Santa Cruz e Centro de Desenvolvimento Tecnológico em Saúde do Instituto Oswaldo Cruz
nicolas.carels@gmail.com

Diego Frias
Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas da Universidade Estadual de Santa Cruz, Bahia, Brazil
diego.cepedi@gmail.com

Abstract

Computational resources have become essential for genome project development. Distributed systems managing complex structures integrating graphical user interfaces, expensive data processing, data mining and large databases, have been proposed. Most consolidated sequencing laboratories have developed their own bioinformatics solutions. However, a portable and scalable system integrating all these aspects is not yet available to the scientific community. In this report, we present the prototype of such a system in open development at http://sourceforge.net/projects/askgene. It allows for the (i) accessibility of data and processes all along the data flow, (ii) data representation and ontology, (iii) workflow tuning, (iv) system architecture and documentation, (v) corporate development, (vi) manual annotation, (vii) bogus data processing, (viii) process parallelization and distribution, (ix) portability and scalability.

Keywords

Gene annotation, Perl, relational database, PostgreSQL, open source

Introduction

Expressed Sequence Tags (ESTs) and Shotgun Sequencing of DNA are fundamental resources for comparative and functional genomics. With the onset of the Massive Parallel Sequencing technology (LEAMON et al., 2007), one may expect a decrease of DNA sequencing cost per nucleotide together with a tremendous increase of sequence flow. The increasing popularity and availability of DNA sequencing is promoting the fast development of the field of computational mathematics applied to biology.

The open source GNU Linux operational system allows for the integration of computational frameworks and tools in a common professional package available to everyone. Together with Linux maturation, we are witnessing a process of large programming language evolution driven by two antagonist objectives: increasing calculation speed and improving language usability.

Bioinformatics handles the genetic codes at DNA, RNA and protein levels, which makes it inherently text oriented. For this reason, text-processing languages like Perl (http://www.perl.org) and Python (http://www.python.org) have been chosen for the development of bioinformatic tools. Perl is a mildly structured language, easy to learn, that enables fast programming of a very large set of applications including Common Gateway Interface (CGI), database and WEB application components. The Perl developer community is very active, contributing voluntarily with extensively tested source codes, which are available to the publicly at on-line repositories. This increases Perl attractiveness and reduces time and developing effort. Most libraries are released in the form of packages called "Perl Modules" to a main repository called CPAN (Comprehensive Perl Archive Network, http://www.cpan.org), the gateway to "All Things Perl" (http://www.cpan.org). One of the most valuable features of CPAN is that the comprehensive index of Perl Modules is fully searchable and maintained up to date. Recently, a bioinformatics oriented library, called BioPerl (http://www.bioperl.org) was included in the CPAN resources. The first contributions started in 2001 and the BioPerl last version is 1.5.2 with more than 200 modules currently available. Perl 5.8.8 is the current version of the Perl programming language, which evolves continuously following hardware and operational systems evolution.

When capillary DNA sequencing machines are used (SMITH et al., 1986), raw DNA sequences are obtained and need to be preprocessed before attempting to interpret their biological functions. In fact, raw sequences could contain sequencing errors and be contaminated with sequence fragments such as from the cloning vector, adapters and poly-A tail, that must be identified and filtered to yield reliable data. The preprocessing operations are frequently done following a sequential workflow known as a bioinformatics pipeline. Such workflow depends on the technology used for cloning and sequencing the DNA fragments and must be customizable by the user.

Bioinformatics pipelines usually integrate computational tools for data acquisition (raw data submission), preprocessing, storage, analysis and mining. However, a particular implementation of a bioinformatics pipeline must be fully customizable and flexible. The system called ASKGene - Automatic Sequence Knowledge Generator - is currently under open development and it is accessible at http://sourceforge.net/projects/askgene. It is written in Perl and will be available soon as a CPAN module. In this article, we describe a prototype of this system for the automation of such basic DNA manipulation.

The DNA sequence pipeline

The workflow that is implemented in the present version of ASKGene performs basically the following 9 operations:

1. Submission of raw sequences, which can be electropherograms of single sequences or electropherograms of a library plate, or even the string of sequences given in FASTA format. This is possible because the submission module was projected for handling heterogeneous input. All submitted sequences are saved in a relational database created for each project. Each database entry is indexed by project, workflow configuration, data source (plate, well, file, etc), submission date, time and user, etc.

2. Electropherogram translation to string and estimation of the quality of each base in the sequence (http://www.phrap.org) (only for electropherogram input). This process is inversely proportional to the probability of sequencing errors.

3. Filtering of low quality regions. The user can define the quality threshold to be used in order to ensure a uniform quality of the data saved to the database. The DNA sequence is processed to extract its sub-sequence corresponding to the given Phred quality threshold. There is always a compromise between the sequence quality and the maximum amount of information that can be retrieved from the electropherograms. Actually, it may occur that a given sequence must be eliminated because of its overall low quality, although, during the similarity search (see step 7), such sequence could produce a hit that implies similarity with a known gene. In this case, a valuable-information-carrying sequence would be eliminated from the database.

4. Filtering of cloning vectors and other sequence contamination. This operation identifies segments within the DNA sequences that could be of vector origin. ASKGene performs screening using UniVec (ftp://ftp.ncbi.nih.gov/pub/UniVec/) that also contains sequences for the adapters and linkers commonly used in the process of cloning cDNA or genomic DNA. This enables the detection of raw sequence contamination with linker fragments. The output of the program is a modified version of the input sequence in which foreign sequence segments are replaced by Xs. The comparison of raw and vector sequences is performed by the program Cross-match (http://www.phrap.org), which is an efficient implementation of the Smith-Waterman-Gotoh algorithm.

5. Filtering of poly-A tail (RNA/cDNA sequencing) and masking of low complexity sub-sequences and interspersed repeats. ASKGene applies the program RepeatMasker (http://www.repeatmasker.org) in order to filter out these sequences. The program output is a sequence in which all identified repeats are masked (replaced by Ns). RepeatMasker also accesses the Repbase database (http://www.girinst.org/repbase/update/index.html) which is a reference database of eukaryotic repetitive DNA. It includes repeat sequences and their descriptions. ASKGene allows users to chose and configure all the filtering steps described above.

6. Sequence assembly. The filtered sequences can be assembled into contiguous sequences (contigs) using the program CAP3 (HUANG et al., 1999); <http://pbil.univ-lyon1.fr/cap3.php>. Sequences that do not overlap with any other are called singlets. This operation has 3 main consequences: (1) it reduces the redundancy in the database, (2) it increases the average size of the sequences by promoting sequence overlap and (3) it increases the sequence quality (reliability) by constructing consensus sequences.

7. Similarity searches by comparing each database entry with other biological sequence repositories for sequence similarities, using local alignment algorithms. The databases selected through the workflow configuration can be those available through the Internet or local repositories. This step, often called "functional annotation" is performed using the BLAST package (Altschul et al., 1990). Potential homologs of the singlets and contigs are searched in 'nr' (the non-redundant section of GenBank), SWISSPROT (http://www.ebi.ac.uk/swissprot/), Pfam (http://pfam.sanger.ac.uk/) and Gene Ontology - GO (http://www.geneontology.org/) databases using the blastx algorithm (http://www.ncbi.nlm.nih.gov/BLAST).

8. Displaying the results of each operation performed in the workflow. A suitable graphical web interface was designed for each case.

The user configures the workflow by eliminating undesired steps and modifying the parameters of the steps included. One can also import new functions and programs to perform additional operations. Thus, the user first builds his customized workflow and then executes it.

The processing flux

The system is made up with 4 basic components: (i) the front-end, (ii) the back-end, (iii) the agent and (iv) the relational database (Figure 1).

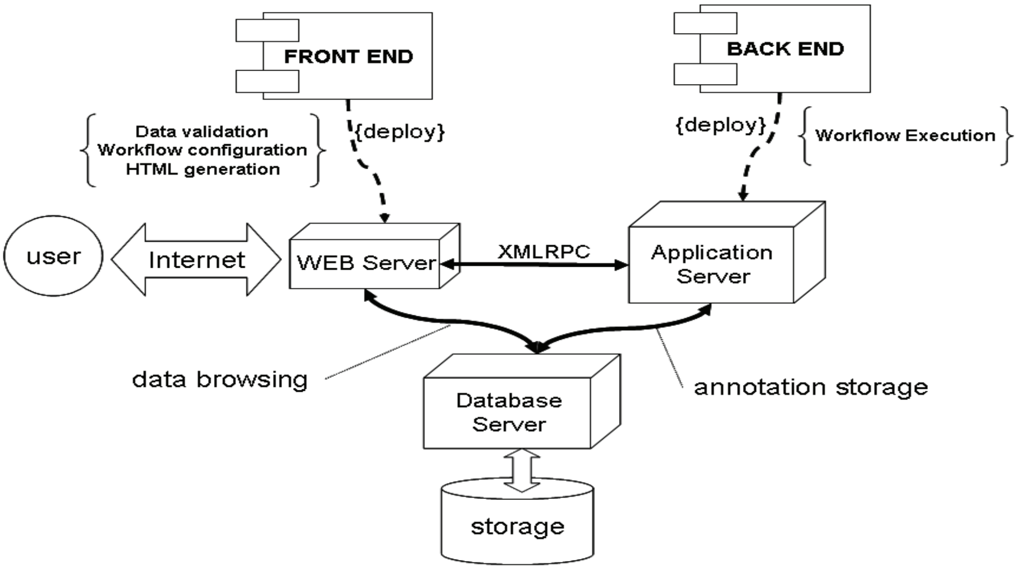

Figure 1 - ASKGene system architecture.

All the results from the workflow execution are stored in a relational database and are managed by the browser agent.

The front-end is the interface between the user and the agent. The agent manages the process flux, executes the workflow and feeds the relational database. It organizes the process stack in order to avoid its overflow. If the request flux exceeds the database processing abilities, the stack size increases and *vice versa*.

The back-end is the interface with the workflow execution and the relational database; it (i) verifies the process that is submitted to the database, (ii) controls its security, (iii) generates a report, (iv) maintains the integrity, (v) ensures the system robustness and (vi) manages the workflow construction. The back-end is responsible for decompressing data and executing the workflow. Both front-end and back-end were set up within the Apache2 HTTP (www.apache.org/) server. This server was further extended by the mod_Perl module, which is far more robust than the test server provided by Catalyst. The Apache server implements the HTTP protocol using the boss/worker parallelism model. The mod_Perl extension embeds a Perl interpreter inside each Apache process and loads the system along with the server, eliminating the start-up overheads of the CGI approach.

The PostgreSQL Database Managing System (www.postgresql.org) was chosen for being a powerful open source Object Oriented programming (OOP) system.

The prototype

Several existing libraries were used, most of them based on the Catalyst Perl module. Catalyst is an open source, object-oriented framework for developing web-based systems. It provides the necessary infrastructure for interacting with web browsers via the Hyper-Text Transfer Protocol (HTTP) using the Model-View-Controller (MVC) design pattern. The prototype's main View component involves Template Toolkit (http://www.template-toolkit.org/), which is a template for filling in generic text models with data, in order to generate the pages in Hyper-Text Markup Language (HTML). The client-side software was developed using the Dojo Javascript Toolkit (http://www.dojotoolkit.org/). This library provides components for building user interfaces as well as an Asynchronous JavaScript and XML (AJAX) implementation.

Compared to the classical strategy of CGI programming, the direct interaction of JavaScript objects in the user web page with Perl objects in the server allows overload saving. When CGI is used as the background technology for web development, the process triggered by the user occurs in 3 steps: (i) loading the system into memory; (ii) processing the user's request; (iii) unloading the system from memory. CGI is quite ineffective for large system management that takes time to initialize (start-up overhead) and is heavily accessed (about 30 simultaneous requests or more).

Upon workflow configuration, the user selects a pipeline to be executed by the system. The system then automatically parses a file with meta-data for database and program locations. The data are also parsed by the system in order to reconstruct the object structures necessary for running the workflow.

Results generated by the selected workflow are stored alongside with a pointer to the workflow meta-data. This allows permanent association of a sequence or a given annotation with the process from which it was issued. New workflow processing steps can be added simply by incorporating a new class to the system and implementing the correct input and output conversion methods.

The DBIx::Class module was used as the application model for data-access via Object-Relational Mapping. In this way, genes, annotations, workflows and processing parameters can be stored in the Relational Database (RDB) and accessed by the system as regular objects. DBIx::Class provides abstractions for traditional RDB facilities such as automatic SQL translation between different Database Management Systems (DBMS).

A project can be subdivided in sections representing libraries corresponding to 96-well-plates used for DNA cloning and sequencing operation. The electropherograms corresponding to these plates are submitted to one or more workflows that are created by the user. These workflows are composted by various processes (jobs) with their own parameters. A job is a workflow instance that has three states: waiting, executing and done. In this way, the workflow storing process can be limited to the storing of their specific pipeline and corresponding history for a given project.

A FIFO (First In First Out) queue is used to control job execution. This queue is also used for process recuperation in case of power failure in the middle of a workflow execution. The results of the sequencing and annotation processes are available through specific documents.

In contrast to the operational database, the analytical database supports analytical processing, *i.e.*, trading intelligence and knowledge finding that are necessary for decision-making, planning and management. The analytical database is a relational database that is used for the storage of statistical information concerning the sequencing and annotation processes. These statistics are for instance: the percentage of accepted reads, average read size, number of contigs, number of hits, etc. These data can be retrieved for a given job, entry, workflow node, and time interval. A summary for each process is also provided after complete workflow execution. The analytical database has been created under the report scheme, which allows the distinction of its table and index space from that of the operational database that is public.

The user-friendly interface for the system development includes (i) automatic code generation, (ii) a simple HTTP server for live testing purposes and (iii) the basic procedures regarding the interaction with the HTTP protocol. This eliminates the need for an expensive server to centralize tests by developers. Every developer can have a copy of the system and run it locally.

Acknowledgments

This work has been supported by the Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) providing fellowships to E. Cardim, W. Reis and N. Carels. We thank Eduardo Costa for the management of NBCGIB computers.

Bibliographic references

ALTSCHUL, S.F. et al. Basic local alignment search tool. **Journal of Molecular Biology**, v.215, p.403-10, 1990.

HUANG, X.; MADAN, A. CAP3: A DNA Sequence Assembly. **Program. Genome Res.**, v.9, p.868-877, 1999.

LEAMON, J.; ROTHBERG, J. Cramming more sequencing reactions onto microreactor chips. **Chemical Reviews**, v.107, p.3367-3376, 2007.

SMITH, L.M. et al. Fluorescence detection in automated DNA sequence analysis. **Nature**, v.321, p.674-679, 1986.

About the authors

Eden Cardoso

Graduated in Computation Science from the State University of Santa Cruz – UESC (Bahia). He also obtained a scholarship of scientific initiation (IC) from the Fundation for the Support to Research from the State of Bahia – FAPESB. Eden is member of the Perl users in Salvador (Salvador Perl Mongers) and coordinator of the Brazilian Perl Society. Eden is also Researcher at the Nucleus of Computational Biology and Management of Biotechnological Information – NBCGIB, UESC.

Wallace Vinicius Oliveira Reis

Graduated in Computation Science from the State University of Santa Cruz – UESC (Bahia). He worked with biological databases and system development for the Web. He also obtained a scholarship of scientific initiation (IC) from the Fundation for the Support to Research from the State of Bahia – FAPESB. Wallace is member of the Perl users in Salvador (Salvador Perl Mongers). Wallace is also Researcher at the Nucleus of Computational Biology and Management of Biotechnological Information – NBCGIB, UESC.