

ASKGene, um sistema para processamento automatizado de DNA

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



Eden Cardim

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



Wallace Reis

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

Resumo

Os recursos computacionais se tornaram essenciais para o desenvolvimento de projetos genoma. Vários sistemas distribuídos que gerenciam estruturas complexas e que integram interfaces gráficas voltadas para o usuário, processamento e mineração de grandes quantidades de dados e bancos de dados volumosos, foram propostos. A maioria dos laboratórios de seqüenciamento já consolidados desenvolveu soluções próprias. Entretanto, um sistema portátil e escalonável, integrando todos esses aspectos, ainda não está disponível para a comunidade científica. Neste estudo, apresentamos um protótipo de tal sistema em curso de desenvolvimento aberto em <http://sourceforge.net/projects/askgene>. Este sistema permite (i) acessibilidade aos dados e processos ao longo de todo o fluxo de dados, (ii) representação de dados e ontologia, (iii) manejo do fluxo de processamento (*workflow*), (iv) arquitetura e documentação do sistema, (v) desenvolvimento corporativo, anotação manual, (vi) processamento de dados corrompidos, (viii) distribuição e paralelização de processos, (ix) portabilidade e escalonabilidade.

Palavras-chave

Anotação de gene, Perl, banco de dados relacional, PostgreSQL, código aberto

Introdução

Expressed Sequence Tags (ESTs – etiquetas de seqüências de RNA mensageiro) e *DNA Shotgun Reads* (seqüências de fragmentos de DNA produzidos ao acaso) são fontes fundamentais para estudos de genômica funcional e comparativa. Com o que está por vir da tecnologia de Seqüenciamento Paralelo Massivo (LEAMON, ROTHBERG 2007), espera-se uma diminuição nos custos de seqüenciamento de DNA por nucleotídeo junto com um enorme aumento no fluxo de seqüências. A crescente popularidade e disponibilidade do seqüenciamento de DNA está promovendo um rápido desenvolvimento no campo da matemática computacional aplicada à Biologia.

O sistema operacional de código aberto GNU Linux permite a integração de arcabouços (*frameworks*) computacionais e ferramentas em um pacote profissional comum disponível para todas as pessoas. Juntamente com o amadurecimento do Linux, nós estamos testemunhando um processo de grande evolução da linguagem de programação, dirigida por dois objetivos antagônicos: aumentar a velocidade de cálculo e melhorar a usabilidade da linguagem.

A Bioinformática lida com os códigos genéticos nos níveis de DNA, RNA e proteína, o que a torna inerentemente orientada por texto. Por esta razão, linguagens de processamento de texto como Perl (<http://www.perl.org>) e Python (<http://www.python.org>) têm sido escolhidas para o desenvolvimento de ferramentas de bioinformática. Perl é uma linguagem moderadamente estruturada, fácil de aprender, que permite a programação rápida de um grande conjunto de aplicações, incluindo CGI (*Common Gateway Interface*), banco de dados e componentes de aplicativos *web*. A comunidade de desenvolvedores em Perl é muito ativa, contribuindo voluntariamente com códigos fontes amplamente testados que estão disponíveis em repositórios públicos *online*. Isto aumenta a atratividade do Perl e reduz esforço e tempo de desenvolvimento. A maioria das bibliotecas está disponível na forma de pacotes (módulos) em um repositório principal chamado CPAN (*Comprehensive Perl Archive Network*, <http://www.cpan.org>) cujo portal providencia acesso a tudo relacionado à linguagem Perl (<http://www.cpan.org>). Uma das características mais valiosas do CPAN é que a lista dos módulos Perl é totalmente acessível para pesquisa e é regularmente atualizada. Recentemente, uma biblioteca direcionada à bioinformática, chamada BioPerl (<http://www.bioperl.org>), foi incluída nos repositórios do CPAN. As primeiras contribuições ocorreram em 2001 e atualmente a última versão do BioPerl é a 1.5.2 com mais de 200 módulos disponíveis. Perl 5.8.8 é a versão atual da linguagem de programação Perl, que evolui continuamente, acompanhando a evolução do *hardware* e dos sistemas operacionais.

As seqüências primárias de DNA obtidas por tecnologia de seqüenciamento capilar (SMITH et al., 1986), precisam ser pré-processadas antes de sua utilização para fins biológicos. Na verdade, seqüências primárias podem conter erros de seqüenciamento e estar contaminadas com fragmentos de seqüências do vetor de clonagem, adaptado-

res ou caudas poli-A que devem ser identificadas e filtradas para que dados confiáveis sejam produzidos. As operações de pré-processamento são freqüentemente feitas seguindo um *workflow* seqüencial conhecido como *pipeline* (canal de processamento). Este *workflow* depende da tecnologia utilizada para clonagem e seqüenciamento de fragmentos de DNA e deve ser personalizável pelo usuário.

Pipelines geralmente integram ferramentas computacionais para aquisição de dados (submissão de dados primários), pré-processamento, armazenamento, análise e exploração. No entanto, uma implementação particular de um *pipeline* deve ser completamente personalizável e flexível. O sistema chamado ASKGene – *Automatic Sequence Knowledge Generator* –, atualmente em desenvolvimento aberto e acessível no endereço <http://sourceforge.net/projects/askgene> foi escrito em Perl e estará disponível em breve como um módulo do CPAN. Neste artigo, descrevemos o protótipo deste sistema para automação de tais manipulações de base de DNA.

O canal de processamento das seqüências de DNA

O *workflow* que está implementado na versão atual do ASKGene executa basicamente as 9 operações seguintes:

1. submissão de seqüências primárias, que podem ser os eletroferogramas de seqüências isoladas ou eletroferogramas de uma placa contendo uma biblioteca, ou até mesmo o conjunto de seqüências fornecidas em formato FASTA. Isto é possível, pois o módulo de submissão foi projetado para lidar com dados de entrada heterogêneos. Todas as seqüências submetidas são armazenadas em um banco de dados relacional criado para cada projeto. O acesso a cada banco de dados é indexado por projeto, configuração do *workflow*, fonte de dados (placa, poço, arquivo, etc.), data de submissão, hora, usuário etc;

2. tradução do eletroferograma para cadeia de caracteres e estimativa da qualidade de cada nucleotídeo na seqüência (<http://www.phrap.org>) (apenas nos casos em que eletroferogramas foram usados como entrada). Este processo é inversamente proporcional à probabilidade de erro de seqüenciamento;

3. filtragem das regiões de baixa qualidade. O usuário pode definir o limiar de qualidade a ser aceito a fim de garantir uma qualidade uniforme dos dados salvos no banco de dados. A seqüência de DNA é processada para extrair sua subseqüência correspondente ao limiar de qualidade Phred especificado, de acordo com um ajuste entre a qualidade e a quantidade de informação que pode ser recuperada das seqüências. Na verdade, pode ocorrer que uma seqüência seja eliminada por apresentar uma qualidade extremamente baixa, embora, durante a busca por similaridade (vide item 7), tal seqüência possa produzir um *hit* que sugira homologia com um gene conhecido. Neste caso, uma seqüência portadora de informação valiosa seria eliminada do banco de dados;

4. filtragem de vetores de clonagem e outros artefatos que contaminem as seqüências. Esta operação identifica segmentos nas seqüências de DNA que podem

pertencer ao vetor de clonagem. O ASKGene executa uma varredura utilizando o UniVec (<ftp://ftp.ncbi.nih.gov/pub/UniVec>) que também contém seqüências para os adaptadores e *linkers* comumente utilizados no processo de clonagem de cDNA ou DNA genômico. Isto permite a detecção de contaminação das seqüências primárias com estes oligonucleotídeos. A saída do programa é uma versão modificada da seqüência de entrada na qual os artefatos de clonagem são substituídos por Xs. A comparação das seqüências primária e do vetor é executada pelo programa Cross-match (<http://www.phrap.org>), uma eficiente implementação do algoritmo Smith-Waterman-Gotoh;

5. filtragem da cauda poli-A (seqüenciamento RNA/cDNA) e mascaramento de seqüências repetitivas e de baixa complexidade. ASKGene emprega o programa RepeatMasker (<http://www.repeatmasker.org>) afim de filtrar essas seqüências. A saída do programa é uma seqüência em que todas as repetições estão mascaradas, isto é, substituídas por Ns. O programa RepeatMasker acessa o banco de dados do Repbase (<http://www.girinst.org/repbase/update/index.html>), que é um banco de referência para DNA repetitivo eucariótico. Este banco contém seqüências repetitivas e suas correspondentes descrições. ASKGene permite que o usuário escolha e configure todas as etapas de filtragem descritas acima;

6. montagem das seqüências. As seqüências filtradas podem ser agrupadas em seqüências contíguas (*contigs*) utilizando o programa CAP3 (HUANG, MADAN 1999; <http://pbil.univ-lyon1.fr/cap3.php>). Seqüências que não se combinam por sobreposição com nenhuma outra são chamadas de *singlets*. Esta operação tem 3 conseqüências: (1) reduz a redundância no banco de dados, através da eliminação da redundância entre as seqüências, (2) aumenta o tamanho médio das seqüências, através do

favorecimento da sobreposição entre as mesmas e (3) aumenta a qualidade das seqüências (confiabilidade) ao conservar a seqüência consenso;

7. busca por homologia, através da comparação de cada entrada no banco de dados do sistema com outros repositórios de seqüências biológicas à procura de similaridade entre as seqüências, com uso de algoritmos de alinhamento local. Os bancos de dados considerados pelo sistema são escolhidos pelo usuário no momento da configuração do *workflow*. Podem ser externos, selecionados via *Internet*, ou repositórios locais. Esta etapa, freqüentemente chamada de “anotação funcional” é executada utilizando o pacote BLAST (ALTSCHUL et al. 1990). A procura por potenciais homólogos aos *singlets* e *contigs* são executadas nos bancos ‘nr’ (a seção não redundante do GenBank), SWISSPROT (<http://www.ebi.ac.uk/swissprot/>), Pfam (<http://pfam.sanger.ac.uk/>) e Gene Ontology - GO (<http://www.geneontology.org/>), utilizando o algoritmo blastx (<http://www.ncbi.nlm.nih.gov/BLAST>);

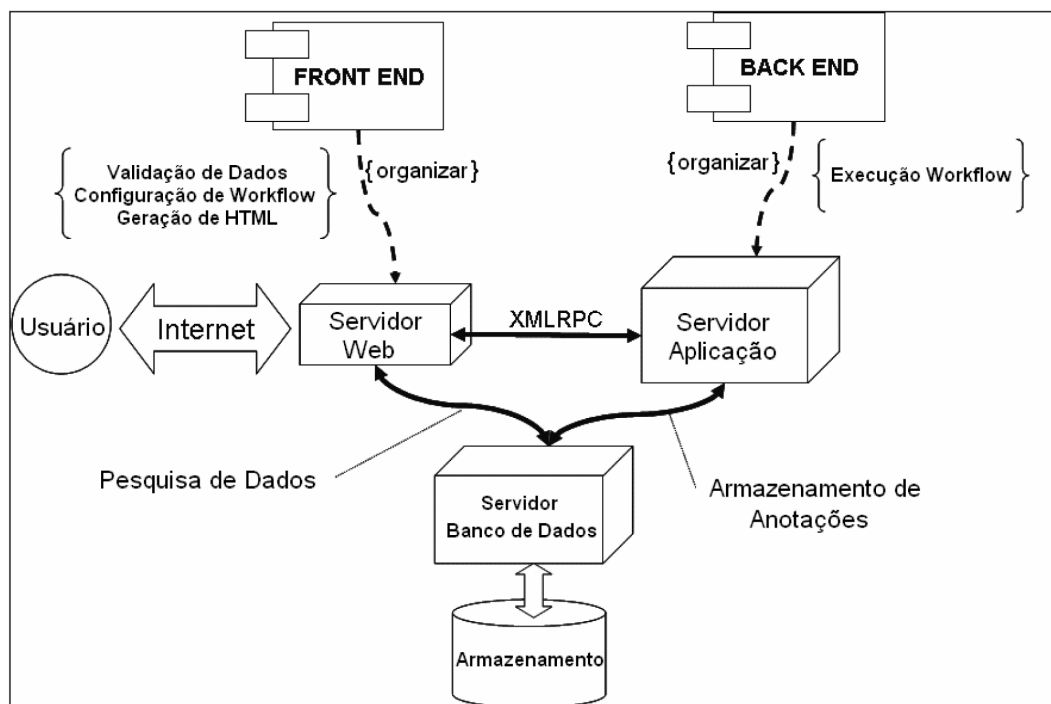
8. exibição dos resultados de cada uma das operações realizadas no *workflow*. Uma interface gráfica *web* apropriada foi desenvolvida para cada caso.

O usuário configura o *workflow* ao eliminar etapas da versão nativa, modificar os parâmetros de configuração das etapas incluídas, e também importar novas funções e programas para executar as operações desejadas. Na verdade, o usuário primeiro constrói o seu *workflow* e então o executa.

O fluxo de processamento

O sistema é formado por 4 componentes básicos: (i) o *front-end*, (ii) o *back-end*, (iii) o agente e (iv) o banco de dados relacional (Figura 1).

Figura 1 – Arquitetura do sistema ASKGene



Todos os resultados da execução do *workflow* são armazenados em um banco de dados relacional e são gerenciados pelo agente do navegador.

O *front-end* é a interface entre o usuário e o agente. O agente gerencia o fluxo de processos, executa o *workflow* e armazena os dados no banco relacional. Ele organiza a pilha de processos a fim de evitar sua sobrecarga. Se o fluxo de solicitação excede a capacidade de processamento do banco de dados, o tamanho da pilha cresce e vice versa.

O *back-end* é a interface com a execução do *workflow* e com o banco de dados relacional; ele (i) verifica o processo que é submetido ao banco de dados, (ii) controla sua segurança, (iii) gera um relatório, (iv) mantém a integridade, (v) garante a robustez do sistema, e (vi) gerencia a construção do *workflow*. O *back-end* é responsável por descompactar os dados e executar o *workflow*. Ambos o *front-end* e o *back-end* foram implementados no servidor Apache2 HTTP (www.apache.org). Este servidor foi posteriormente estendido pelo módulo *mod_Perl*, que é muito mais robusto do que o servidor de testes fornecido pelo Catalyst. O servidor Apache implementa o protocolo HTTP usando o modelo de paralelismo *boss/worker* (chefe/trabalhador). A extensão *mod_Perl* embute um interpretador Perl dentro de cada processo do Apache e carrega o sistema com o servidor, eliminando as sobrecargas de inicialização da abordagem CGI.

O Sistema de Gerenciamento de Banco de Dados PostgreSQL (www.postgresql.org) foi escolhido por ser um poderoso sistema de programação orientado a objetos (OOP - *Object Oriented Programming*) de código aberto.

O protótipo

Nós utilizamos várias bibliotecas existentes, a maioria delas baseadas no módulo Catalyst do Perl. Catalyst é uma estrutura orientada a objetos de código aberto para desenvolvimento de sistemas *online*. Ele fornece a infra-estrutura necessária para interação com navegadores *web* via HTTP (*Hyper-Text Transfer Protocol*) usando o padrão de estilo *Model-View-Controller* (MVC). Os principais componentes *View* do protótipo incluem o *Template Toolkit* (<http://www.template-toolkit.org/>), que é um molde para preenchimento de textos genéricos com dados, a fim de gerar as páginas em HTML (*Hyper-Text Markup Language*). O programa para gerenciar o lado do usuário (*client*) foi desenvolvido utilizando o *Dojo JavaScript Toolkit* (<http://www.dojotoolkit.org>). Esta biblioteca fornece componentes para construção de interfaces a usuários assim como uma implementação do *Asynchronous JavaScript* e XML (AJAX).

Comparada à estratégia clássica de programação CGI, a interação direta de objetos *JavaScript* na página *web* do usuário com objetos Perl no servidor economiza sua sobrecarga. Quando CGI é usado como tecnologia de base para o desenvolvimento *web*, o processamento disparado pela solicitação do usuário ocorre em 3 etapas: (i) carregamento do sistema na memória; (ii) processamento da solicitação do usuário; (iii) descar-

regamento do sistema da memória. CGI é bastante ineficaz para o manejo de sistemas grandes que levam muito tempo para iniciar (sobrecarga de inicialização) e são altamente acessados (cerca de 30 solicitações simultâneas ou mais).

Na hora de configurar o *workflow*, o usuário seleciona um *pipeline* para ser executado pelo sistema. De acordo com a configuração selecionada pelo usuário, o sistema automaticamente analisa um arquivo de metadados para localizar os programas e o banco de dados. Os dados também são formatados pelo sistema com a finalidade de reconstruir a estrutura de objeto necessária para executar o *workflow*.

Os resultados gerados pelo *workflow* selecionado são armazenados junto com um ponteiro para os metadados do *workflow*. Isto permite a associação permanente de uma seqüência ou de uma determinada anotação com o processo pelo qual ela foi emitida. Novas etapas de processamento pelo *workflow* podem ser adicionadas simplesmente incorporando uma nova classe ao sistema e implementando os métodos de conversão de entrada e saída corretos.

O módulo *DBIx::Class* foi utilizado como o modelo de aplicativo para o acesso a informações via mapeamento relacional dos objetos (*Object Relational Mapping*). Desta forma, genes, anotações, *workflows* e parâmetros de processamento podem ser armazenados em um banco de dados relacional e acessados pelo sistema como objetos regulares. O *DBIx::Class* fornece abstrações para instrumentos tradicionais de banco de dados relacional, tais como a tradução automática de SQL entre diferentes Sistemas de Gerenciamento de Banco de Dados (DBMS - *Database Management System*).

Um projeto pode ser subdividido em seções que representem bibliotecas que correspondam a placas de 96 poços usadas nas operações de clonagem e seqüenciamento de DNA. Os eletroferogramas correspondentes a estas bibliotecas são submetidas para um ou mais *workflows* que são criados pelo usuário. Estes *workflows* são compostos por vários processos (*jobs*) com seus próprios parâmetros. Um processo (*job*) é uma instância do *workflow* que tem três estados: espera, em execução e concluído. Desta forma, o processo de armazenamento do *workflow* pode ser limitado ao armazenamento de seu *pipeline* específico e do histórico correspondente ao projeto em questão.

Uma fila FIFO (First In First Out) é utilizada para controlar a execução dos processos. Esta fila também é utilizada para recuperação de processos em caso de falta de energia elétrica durante a execução do *workflow*. Os resultados dos processos de seqüenciamento e anotação são armazenados em documentos específicos.

Contrastando com o banco de dados de operação, o banco de dados analítico suporta o processamento analítico, *i.e.*, inteligência de negócios e busca de conhecimento que são necessários para a tomada de decisões, planejamento e gerenciamento. O banco de dados analítico é um banco de dados relacional que é utilizado para o armazenamento de informações estatísticas referentes

aos processos de anotação e seqüenciamento. Estas estatísticas são, por exemplo: a porcentagem de seqüências primárias aceitas, tamanho médio das seqüências, número de *contigs*, número de *hits*, etc. Estes dados podem ser obtidos para um processo particular, entrada, nó do *workflow* e intervalo de tempo. Um resumo de cada processo também é fornecido após a completa execução do *workflow*. O banco de dados analítico foi criado sob o esquema de relatório, que permite a distinção de sua tabela e espaço de indexação em relação àqueles do banco de dados de operação que é público.

A interface amigável para o desenvolvimento do sistema inclui (i) geração automática de código, (ii) um servidor HTTP simples para a execução de testes em tempo real, e (iii) os procedimentos básicos acerca da interação com o protocolo HTTP. Isto elimina a necessidade de um servidor caro para centralizar os testes dos desenvolvedores. Cada desenvolvedor pode ter uma cópia do sistema e executá-lo localmente.

Agradecimentos


Este trabalho recebeu apoio da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB), que forneceu bolsas de estudo à E. Cardim, W. Reis e N. Carels. Nós agradecemos ao Eduardo Costa pelo gerenciamento dos computadores do NBCGIB.

Referências bibliográficas

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 Research**, 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. 

Sobre os autores

Eden Cardoso Cardim

É bacharel em Ciência da Computação pela Universidade Estadual de Santa Cruz - UESC (Bahia). Também foi bolsista de Iniciação Científica (IC) pela Fundação de Amparo à Pesquisa do Estado da Bahia – FAPESB. Eden é membro do grupo de usuários Perl de Salvador, Salvador Perl Mongers e Coordenador da Sociedade Perl do Brasil. Pesquisador do Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas – NBCGIB, UESC.

Wallace Vinicius Oliveira Reis

É bacharel em Ciência da Computação pela Universidade Estadual de Santa Cruz - UESC (Bahia) e já trabalhou com banco de dados biológicos e desenvolvimento de sistemas baseado na Web. Também foi bolsista de Iniciação Científica (IC) pela Fundação de Amparo à Pesquisa do Estado da Bahia - FAPESB - trabalhando inicialmente no Laboratório de Bioinformática da UESC - LABBI - e posteriormente no Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas – NBCGIB, UESC.