BioRubyにおける高速なBLAST結果処理機能の実装

Implementation of Fast BLAST output parser in BioRuby

○後藤 直久 Naohisa Goto　安永 照雄 Teruo Yasunaga

大阪大学 遺伝情報実験センター ゲノム情報解析分野
Genome Information Research Center, Osaka Univ.

Abstract

BioRuby is an open-source project which aims to provide a reusable library for biological tasks for the Ruby language. Ruby is an interpreted object-oriented scripting language with a simple and powerful syntax and native object-oriented programming support. BioRuby provides many of typical bioinformatics tasks such as manipulating DNA and protein sequences, retrieval from databases, parsing results of analysis software, and so on. By using BioRuby, we can easily and quickly write programs of bioinformatics analysis. BioRuby is a available as free software and can be downloaded at http://bioruby.org/.

In this poster, we are reporting about implementation of fast BLAST result parser in BioRuby. When analyzing BLAST results, we often write small scripts in Perl, Ruby, Python, Java, and so on. The size of BLAST result output tends to become too large because of the increasing sequence database size in recent year. So, speeding up of BLAST result parsing is very important. However, there have been few programs or libraries which can be easily used under Ruby scripts.

Therefore, we implemented fast BLAST result parser for BioRuby. For fast parsing, we took 'lazy evaluation' technique. We also used strscan, a fast string scanner library for Ruby. As the result, the running spped of it was 5-20 fold faster than BioPerl's parser.

The parser can parse default (-m 0 option) output of NCBI BLAST, including PSI/PHI-BLAST. It only requires Ruby (1,8.0 or later) and does not require any special extensions. It is available with the BioRuby distribution.

要旨

BioRubyはバイオインフォマティクスに必要な機能や環境をオブジェクト指向スクリプト言語Rubyを用いて統合的に実装したライブラリである。塩基・アミノ酸配列の処理や解析、公共データベースのデータ処理、各種解析ソフトウェアの結果処理等に必要な機能を備えており、生物学的解析を行うスクリプトを短時間で容易に書くことができる。BioRubyはフリーソフトウェアであり、http://bioruby.org からダウンロードできる。

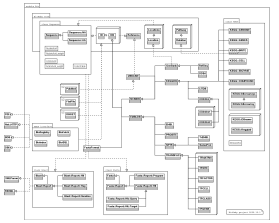
今回は新たに実装した高速なBLASTパーサ(BLAST出力を読み込み解釈する機能)について報告する。PerlやRubyなどのスクリプト言語を用いてホモロジー検索ソフトウェアBLASTの実行結果を読み込み、結果の整理や解析などの後処理を加えることは広く行われている。近年のデータベース容量の増加に伴い、BLAST実行結果の出力サイズは大きくなることが多い。必然的に後処理にも長時間を要することが多くなり、高速化の重要性が高まってきている。しかし、BLAST自体は並列化などにより年々高速化されているが、実行結果の後処理に関しては高速化の検討がまだ十分になされていないことが多い。

そこで今回、高速性を重視したBLASTパーサをBioRubyに実装した。高速化のため、BLAST結果出力を大雑把に分解した後で要求された部分だけを詳細に解釈する遅延評価を導入した。他にも、Ruby標準の高速文字列スキャナを使用するなど様々な高速化の工夫を行った。その結果、広く使われているBioPerlのBLASTパーサと比較すると、12分53秒かかった処理が33秒で完了したなど、処理の内容により異なるが、5倍から20倍以上の処理速度を示した。

このパーサはNCBI BLASTのデフォルト出力(-m 0 オプション)を対象とし、通常のBLASTだけでなくPSI/PHI-BLASTにも対応している。外部拡張モジュールを使用せずRuby(1.8.0以降)標準の機能だけを使用しているため、BioRubyをインストールするだけで利用可能である。

（注）英文と和文の要旨は細部の内容が異なります。

BioRuby



BioRubyプロジェクトは、バイオインフォマティクスに必要な機能や環境を、国産のオブジェクト指向スクリプト言語 Rubyを用いて統合的に実装することを目標にしたオープンソースプロジェクトです。Rubyによるバイオインフォマティクス・生命情報解析用のクラスライブラリとこれを利用したツール類を開発・提供しています。

BioRubyの主な機能

- 塩基配列・アミノ酸配列の操作
  - 翻訳, スプライシング, 検索, …
- 解析ソフトウェアによる解析の支援
  - BLAST, FASTA, HMMER, CLUSTAL W, PSORT, …
    - スクリプト内部からの呼び出し
    - 結果の処理・解析
- データベースのデータ読み込み
  - GenBank, DDBJ, EMBL, SwissProt, KEGG, Prosite, TRANSFAC, AAindex, PDB, PIR, FANTOM, GO, …
  - データ形式の自動認識も可能
- ファイルやインターネットからのデータ取得
  - BioFetch, BioSQL, Flatfile Indexing, DAS, KEGG::API, …
- グラフ, 2項関係, 文献データなど
  - Bio::Pathway, Relation, Reference, MEDLINE

BioRuby Project

http://bioruby.org/

問い合わせ先: staff@bioruby.org

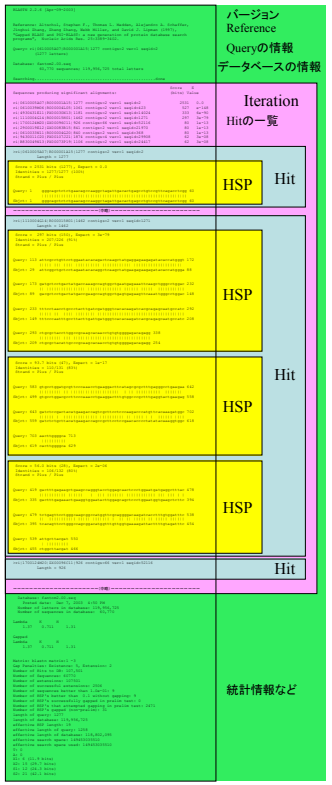
STAFF

片山俊明 – k@bioruby.org
（プロジェクトリーダー）
中尾光輝 – n@bioruby.org
川島秀一 – s@bioruby.org
後藤直久 – ng@bioruby.org

※BioRubyはオープンなプロジェクトです。
いつでも誰でも開発に参加できます。

BLAST結果出力の構造

＜NCBI BLASTのデフォルト出力 (オプション無し、または -m 0 オプション) の例＞



BioRubyのクラス

Bio::Blast::Default::Report クラス

BLAST結果全体。Iterationクラスのインスタンスを内部に保持。

Bio::Blast::Default::Report::Iteration クラス

PSI-BLASTの繰り返し検索1回分の結果を格納するクラスだが、ノーマルのBLASTでも検索結果の格納に使用。Hitクラスのインスタンスを内部に保持。

Bio::Blast::Default::Report::Hit クラス

検索にヒットした配列に関する情報。HSPクラスのインスタンスを内部に保持。

Bio::Blast::Default::Report::HSP クラス

HSP (High-scoring Segment Pair) に関する情報を格納。BLASTによるホモロジー検索結果の最小単位。

※ HSP
High-scoring Segment Pair の略。閾値を超える類似性が検出された部分配列のペア。

サンプルプログラム

各HSPについて、クエリー, ヒットした配列の名前, アライメント長, e-value, ビットスコアをタブ区切りで表示するサンプルプログラム。

```
#!/usr/bin/env ruby
require 'bio'

ff = Bio::FlatFile.auto(ARGF)
print [ 'Query', 'Subject', 'AlignLen', 'eValue', 'BitScore' ].join("¥t"), "¥n"

ff.each do |r|
  qdef = r.query_def.split[0]
  r.each_hit do |hit|
    hdef = hit.definition.split[0]
    hit.each do |hsp|
      alen   = hsp.align_len
      evalue = hsp.evalue
      bscore = hsp.bit_score
      print [ qdef, hdef, alen, evalue, bscore ].join("¥t"), "¥n"
    end
  end
end

ff.close
```

Bio::Blast::Default::Report は Bio::FlatFile によるファイル形式の自動判別に対応している。このため、上のサンプルプログラムではファイル形式（データのクラス, 今回は Bio::Blast::Default::Report）を指定していない。

高速化の工夫

遅延評価 (lazy evaluation): 必要になったときに初めて処理を行う

- 初期化時: BLAST結果をHSPなどの単位で大雑把に分割する。その後、手を加えずそのまま保持する。
- メソッド呼出時: 必要な部分について演算（パース処理）を行う。一度演算した値は記憶し、二度目以降はその値を返す。

BLASTのバージョンや詳細な統計情報は必要としない場合も多い。また、スコアやe-valueが高いヒットについてのみ処理を行うなど、一部のHitやHSPのみを使用し、残りのデータは使用しない場合も多い。このような場合は遅延評価が特に有効である。

高速な文字列スキャナライブラリ strscan の使用

strscan（青木峰郎氏作, Ruby 1.8.0以降に標準添付）

機能比較

	BioRuby (0.5.3)	BioPerl (1.2.1)	Zerg* (1.0.3)
使用言語	Ruby	Perl	C (Perlの拡張ライブラリが存在)
NCBI BLAST(BLASTN/BLASTP/BLASTX/TBLASTN/TBLASTX)対応	○	○	○ (一部の統計情報に未対応)
HSPのアライメント取得	○	○	×
PSI-BLAST対応	○	○	×
WU-BLAST対応	○ (一部の統計情報に未対応)	○	×

*Paquola, A.C.M, et al. (2003), Zerg: a very fast BLAST parser library, *Bioinformatics*, **19**,1035-1036.

速度比較

PentiumIII 1.0GHz, メモリ1GB, HDD 27GB, 100Mbps Ethernet, Linux 2.4.18 というスペックのマシン上でベンチマークプログラム* を10回動かしたときの平均所要時間と処理速度およびBioPerlを基準とした速度の比率を示した。

* http://bioinfo.iq.usp.br/zerg/zerg_benchmarks_1.0.tar.gz
ただしBioRubyに関しては同等機能のプログラムを独自に作成した。
ベンチマークに使用したデータは以下のとおりである。
BLASTN: 104,921,408バイト, 8014エントリ
BLASTX: 104,858,552バイト, 16013エントリ

	BLASTN 所要時間(s)	BLASTN S.D.	BLASTN 速度(MB/s)	BLASTN 速度比	BLASTX 所要時間(s)	BLASTX S.D.	BLASTX 速度(MB/s)	BLASTX 速度比
BioRuby (Ruby1.8.0)	35.325	0.032	2.83	21.3	44.821	0.084	2.23	23.9
BioRuby (Ruby1.6.7)	49.724	0.048	2.01	15.1	79.857	0.083	1.25	13.4
BioPerl (Perl5.6.1)	751.067	2.915	0.133	1.0	1070.301	5.098	0.0934	1.0
Zerg-C	2.437	0.002	41.1	308	2.685	0.001	37.2	399
Zerg-Perl	2.605	0.002	38.4	288	2.977	0.002	33.6	360
Zerg-Perl2 (Perlオブジェクトを作成)	36.687	0.051	2.73	20.5	57.675	0.222	1.73	18.6

まとめ

BioRubyのBLASTデフォルト出力パーサは、最速ではないものの、スクリプト言語標準搭載の機能だけを使用して書かれたパーサとしては非常に高速である。BLAST結果処理にBioRubyを使用すると、スクリプト言語の柔軟性と処理の高速性の両方を得ることが可能であり、解析作業の効率を上げることが可能であると期待される。