Programming with BLAST

BLAST Scripting

May 26, 2005 *NCBI / NLM / NIH*

What is BLAST Scripting?

- Running BLAST non-interactively
- The three options:
  - BLAST URL API (posting URLs to NCBI)
  - BLAST network client (blastcl3)
  - Standalone BLAST

Why script BLAST?

- To automate searches
- To incorporate BLAST searches into existing software
- To create a customized BLAST interface
- To process large numbers of queries
- To annotate genomic DNA

Which tool to use?

	URL API	Blastcl3	Standalone	wwwblast
Database	NCBI	NCBI	Local, custom	Local, custom
CPU	NCBI	NCBI	Local	Local
Programs	Basic 5+	Basic 5+	All	All
Interface	via script	CMD line	CMD line	GUI/script
Batch?	Manual	Automatic	Automatic	Manual/script

Advantages of the URL API

- cron jobs: one process can control many BLAST calls
- Diverse batches: retrieval and/or formatting parameters can be varied for constant queries
- Diverse batches: one process can control multiple different BLAST programs
- Universal interface: no NCBI software required!
- Easiest way to construct BLAST links within documents
- Allows the greatest flexibility in data pipelines

BLAST URL API

starring your friend and mine…

Blast.cgi

http://www.ncbi.nlm.nih.gov/blast/Blast.cgi

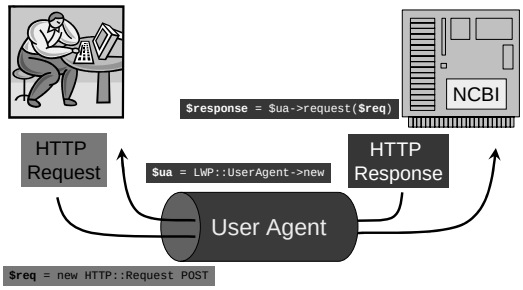
A Sample URL

http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?CMD=Put
&PROGRAM=blastn&DATABASE=nr&FILTER=L&QUERY=AF123456

CMD	Put : submit a query
PROGRAM	blastn : run BLASTn
DATABASE	nr : search against nr
FILTER	L : turn low complexity filtering on
QUERY	AF123456 : accession, GI, or FASTA

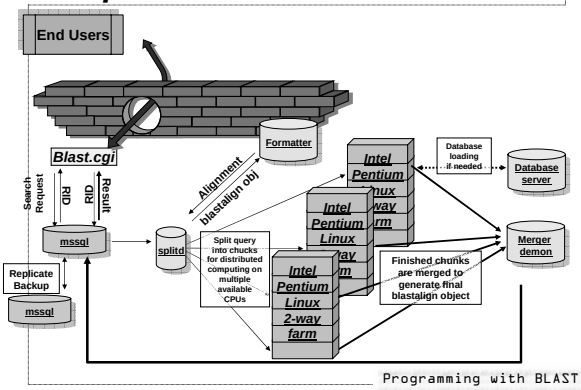
An interim update to BLAST URLAPI, still being reviewed, is at:
http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/node_0.html

Posting a URL



For Perl, you need libwww-perl-5.800

splitd NCBI BLAST Server



Using the URL API

- Read a FASTA file (query sequences)
  construct URL with query and parameters
- Post a 'PUT' URL
  retrieve the result, parse out the RID, construct URL
- Post a 'GET' URL
  if done, write the results to a file or to RAM
- Parse the results
  generate an output file

Issues of Timing

- Each RID requires at least 2 URLs: 1 Put and 1 Get
- Our recommendation: post with delays of >= 3 sec
  - this equals 28,800 URLs per day
- How often do you post a Get URL?
- For 100 RIDs, if you check every minute…
  - 100 URLs/minute x 10 minutes = 1000 URLs in 10 minutes
- But…what if there's a problem for 6 hrs, 24 hrs??
  - 100 URLs/minute x 360 minutes = 36,000 URLs per day!
  - 100 URLs/minute x 1440 minutes = 144,000 URLs per day!

Your script must check for errors!!

blast_single.pl

Performs a single BLAST search

```
perl blast_single.pl input_file
```

Subroutines:

- read_params; read_fasta
- put_url (&CMD=Put)
- get_url (&CMD=Get)
- parse_tabular

Input File – Single Query

```
input_file|danio_1402.txt
program|blastn
database|nr
filter|L
entrez_query|srcdb+refseq[prop]
```

```
Blast.cgi?CMD=Put&PROGRAM=blastn&DATABASE=nr&FILTER=L
&ENTREZ_QUERY=srcdb+refseq[prop]&QUERY=<danio_1402.txt>
```


sub read_params

```
my ($param, $value);
my %params;

#check for correct command line syntax
if ($#ARGV != 0) { die "Usage: [eutil].pl input_file\n"; }

#read input parameter file
open(INPUT, "<$ARGV[0]") || die "Aborting. Can't open $ARGV[0]\n";

while (<INPUT>) {

   chomp;
   ($param, $value) = split(/\|/);
   $params{$param} = $value;

}

close INPUT;

return (%params);
```

sub read_fasta

```
my %params = @_;
my $limit = 1;
my $seq;
my $c;
my $source_file = $params{'input_file'};

$source_file = $path . $source_file;

open (INPUT, "$source_file") || die "Can't open $source_file!\n";

while (<INPUT>) {

  $c++ if (/^>/);
  last if ($c > $limit);

  next if (/^\s*#/); #ignore comment lines
  next if (/^\s*$/); #ignore blank lines
  $seq .= uri_escape($_);

}

close(INPUT);

return($seq);
```


sub put_url – Part 1

```
my %params = @_;
my $ua = LWP::UserAgent->new;
my ($req, $response, $html, $rid);
my $search_args;

$search_args = "CMD=Put&PROGRAM=$params{'program'}";
$search_args .= "&DATABASE=$params{'database'}&EXPECT=$params{'expect'}";
$search_args .= "&FILTER=$params{'filter'}&HITLIST_SIZE=$params{'hitlist_size'}";
$search_args .= "&QUERY=$params{'query'}&ENTREZ_QUERY=$params{'entrez_query'}";
$search_args .= "&QUERY_FROM=$params{'query_from'}&QUERY_TO=$params{'query_to'}";
$search_args .= "&WORD_SIZE=$params{'word_size'}";

#submit search
$req = new HTTP::Request POST => 'http://www.ncbi.nlm.nih.gov/blast/Blast.cgi';
$req->content_type('application/x-www-form-urlencoded');
$req->content($search_args);

$response = $ua->request($req);
```


Output of a PUT URL

Search in Progress

```
<HTML>
<HEAD>
...
</HEAD>
<BODY>
...
<!--
QBlastInfoBegin
       RID = 1117036222-2037-19313847822.BLASTQ2
QBlastInfoEnd
--></form>
</body>
</html>
```

sub put_url – Part 2

```
if (!$response->is_success) {
    print $response->error_as_HTML;
    die;
}

$html = $response->content;
$html =~ /RID = (.+)/;
$rid = $1;
print "RID: $rid\n" if ($verbose);

return $rid;
```


Output of a GET URL

Search in Progress

```
<HTML>
<HEAD>
</HEAD>
<BODY>
...
<!--
QBlastInfoBegin
       Status=WAITING
QBlastInfoEnd
--><p>
timing report
```

Completed Search

```
<HTML>
<HEAD>
</HEAD>
<BODY>
...
<!--
QBlastInfoBegin
       Status=READY
QBlastInfoEnd
--><p><pre>
blast output
```


sub get_url – Part 1

```
$delay = 15;
$maxdelay = 4*4*4*$delay;

   $done = 0;
   $status = 0;
   $c = 0;
   until ($done) {

      unless ($status) {
        $retrieve_args = "CMD=Get&ALIGNMENT_VIEW=Tabular&RID=$rid";
        $ret_url = 'http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?' . $retrieve_args;

        $results = get($ret_url);
        if ($results =~ /READY/) {
          $done = 1;
        }
        elsif (($results =~ /ERROR/) || ($results =~ /Error/)) {
          print "$results\n";
          exit;
        }
      }
      ...
```

sub get_url – Part 2

Check the RID periodically, but not too often!

15 sec → 1 min → 4 min → 16 min

```
 $c++;
     if ($c >= 10) {
        $c = 0;
        $delay *= 4;
     }
     $delay = $maxdelay if ($delay > $maxdelay);
     sleep($delay) unless ($done);
   }

return $results;
```


main

```
my ($query, $rid);
my %params;
my $results;
my $verbose = 1;
my $path = "/home/sayers/blast_powertools/";

die "Usage: single_query.pl source_file\n" unless ($#ARGV == 0);

1 %params = read_params();

2 $params{'query'} = read_fasta(%params);

3 $rid = put_url(%params);

4 $results = get_url($rid);

5 print "$results\n";
```


Blast_API.pm

Perl module containing the following subroutines:

- read_params; read_fasta
- put_url (&CMD=Put)
- get_all_urls (&CMD=Get)
- parse_tabular
- execute
- get_seqs

blast_api.pl

```
   die "Usage: blast_api.pl input_file\n" unless ($#ARGV == 0);

   # read and encode the queries

1  %params = read_params();

2  @queries = read_fasta(%params);
   $nseq = $#queries;

   for ($q=0; $q <= $nseq; $q++) {
      $params{'query'} = $queries[$q];
3     $rid[$q] = put_url(%params);
      sleep($post_delay);
   }

   #get results

4  @results = get_all_urls(@rid);

5  parse_tabular($params{‘output_file’}, $results);
```


Input File – Multiple Queries

```
input_file|danio_1402.txt
output_file|danio.out
program|blastn
database|nr
filter|L
entrez_query|srcdb+refseq[prop]
limit|4
```

Blast.cgi?CMD=Put&PROGRAM=blastn&DATABASE=nr&FILTER=L &ENTREZ_QUERY=srcdb+refseq[prop]&QUERY=<danio_1402.txt>

To query all sequences, set limit = 0

get_all_urls – Multiple Queries

```
until ($done) {

   for ($q=0; $q <= $#rid; $q++) {

     unless ($status[$q]) {
        $retrieve_args = "CMD=Get&ALIGNMENT_VIEW=Tabular&RID=$rid[$q]";
        $ret_url = 'http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?'.$retrieve_args;

        $results = get($ret_url);
        if ($results =~ /READY/) {
         $status[$q] = 1;
         $output[$q] = $results;
        }
        elsif (($results =~ /ERROR/) || ($results =~ /Error/)) {
          print "$results\n";
          exit;
        }
     }
   }
    $done = check_done($#rid, @status);
```

sub parse_tabular

```
 @output = split(/^/, $results);

foreach (@output) {

   if (/Query:/) {
      print OUTPUT "====================\n";
      print OUTPUT "$_\n";
      print OUTPUT "====================\n";
   }

   if (/^gi\|/) {
      chomp;
      push (@data, $_);
   }

}

foreach $hit (@data) {

   @fields = split(/\s+/,$hit);
   print OUTPUT "$fields[1]\t$fields[10]\t$fields[11]\t$fields[2]\t$fields[3]\t";
   print OUTPUT "$fields[6]-$fields[7] <=> $fields[8]-$fields[9]\n";

}
```


Output of parse_tabular

subject | e-value | % id | query start-stop

```
gi|47085732|ref|NM_212962.1|  6e-16  91.7  87.78  90  1657-1746 <=> 1529-1618
```

bit score | alignment length | subject start-stop

Scripting blastcl3 and Standalone

- It's elementary, my dear Watson
- Assign the command to a string
- Execute the string as a system call
- The real benefits: Timing and Parsing

sub execute

syntax: $output = execute($command)

```
sub execute {

my $command = $_[0];
my $output;

open (COM, "$command|");

while (<COM>) {

   $output .= $_;

}

close COM;

return $output;

}
```


sub get_seqs

- Uses the E-Utility EFetch to download FASTA sequences
- syntax: get_seqs($query, $db, $outfile)
  - $query = Entrez query
  - $db = ‘nucleotide’ or ‘protein’
  - $outfile = name of file for output FASTA

Hands-On Session

Set up and perform sample searches

blast_api.pl

Blast_API.pm

Automate database formatting and searching

auto_blast.pl