Running Blast on a Linux Cluster

Ray Hookway
High Performance Computing Expertise Center
Life and Materials Science
June 24, 2003
Motivation

blastall –p blastx –i NC_003070.fna –d nr
The results


Query= AZ-UA-13 NC_003070.3 Large sequence 2[D (30,028,691 letters)

Database: All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF 1,230,998 sequences; 391,609,117 total letters

Searching..................................................................................................................................................
The results (cont)

........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
done

***** No hits found *****

Database: All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF
        Posted date: Nov 8, 2002  1:16 AM
        Number of letters in database: 391,609,117
        Number of sequences in database: 1,230,998

Lambda  K  H
0.318  0.135  0.401
Clusters
Hardware

- **Compute node**
  - IA32 will do (for blast)
  - Blades increase density
  - SMPs are easier to manage and reduce memory cost/processor
  - Want enough memory to hold database
- **Storage**
  - Disk performance important for blastn
  - Not as important for translated versions of blast
- **File server**
  - NAS will do
- **Interconnect**
  - 100 Mb ethernet will do
Software

• Linux
• Management software
• Queuing software (e.g., LSF, PBS, etc.)
  – Control access to cluster
  – Support sequencing of operations
  – Support resource allocation
  – Enhanced reliability
• PVM/MPI
  – Creates parallel jobs – can introduce scheduling problems
• Middleware (e.g., TurboBlast)
• Blast
  – NCBI Blast, WUBlast
Does the database need to be local?

• In general – no
  – Versions of blast which do translations are generally compute bound
  – Blastn can be very disk intensive
  – Keep the database in memory as much as possible
    • This requires smart scheduling

• Usually there isn’t a problem distributing databases within a cluster, however some care is required to maintain a consistent set of databases
How to break up a blast job

blastall –p blastp –o ouptut –i input –d database

foreach query sequence {
    search db;
    print results;
}
Method 1 – Break up input

- Easy to do if the input contains multiple sequences
  - Break input into n chunks
  - Run each chunk independently using the original command line but replacing input with chunk
    
    bsub blastall -p blastx -o output_n -i chunk_n -d database
  - Concatenate the results

    cat output1 output2 ... outputn > output

- Don’t need to worry about where the job is run
- Need to pay attention to the size of individual jobs
- What if input does not contain multiple sequences?
## Query size matters

<table>
<thead>
<tr>
<th></th>
<th>1000 x 1 proc</th>
<th>1000 x 32 proc</th>
<th>8 x 8 proc</th>
<th>32 x 32 proc</th>
</tr>
</thead>
<tbody>
<tr>
<td>blastn</td>
<td>2106</td>
<td>528</td>
<td>277</td>
<td>89</td>
</tr>
<tr>
<td>blastx</td>
<td>8540</td>
<td>687</td>
<td>1210</td>
<td>335</td>
</tr>
<tr>
<td>tblastx</td>
<td>18843</td>
<td>1019</td>
<td>2572</td>
<td>680</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Raw</th>
<th>Scaled</th>
</tr>
</thead>
<tbody>
<tr>
<td>blastn</td>
<td>1 3.988636364</td>
<td>7.602888087</td>
</tr>
<tr>
<td>blastx</td>
<td>1 12.43085881</td>
<td>7.05785124</td>
</tr>
<tr>
<td>tblastx</td>
<td>1 18.49165849</td>
<td>7.326205288</td>
</tr>
</tbody>
</table>
Method 2 – Break up the database

- Break up the database instead of input, again on sequence boundaries
  - Break database into multiple chunks
    - `formatdb -v 1 -i database -p T`
  - Run blast for each chunk using original command line with chunk substituted for database and `-z` switch to set effective database size
    - `bsub blastall -p blastx -o output_n -z effective_db_size \`
      -i input -d database.n`

- Merging output is now more complicated
  - Must gather results for a given query from multiple output files
  - Can concatenate results for a given query
A closer look at the output

BLASTX 2.2.6 [Apr-09-2003]


Query= gi|22330780|ref|NC_003070.3| Arabidopsis thaliana chromosome 1, complete sequence
(30,028,691 letters)

Database: nr1
181,998 sequences; 58,177,152 total letters

Searching..................................................done

<table>
<thead>
<tr>
<th>Score</th>
<th>E Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gi</td>
<td>25518457</td>
</tr>
<tr>
<td>gi</td>
<td>27476100</td>
</tr>
<tr>
<td>gi</td>
<td>4091810</td>
</tr>
<tr>
<td>gi</td>
<td>25406711</td>
</tr>
<tr>
<td>gi</td>
<td>15223294</td>
</tr>
<tr>
<td>gi</td>
<td>3551960</td>
</tr>
<tr>
<td>gi</td>
<td>6715638</td>
</tr>
<tr>
<td>gi</td>
<td>15225767</td>
</tr>
<tr>
<td>gi</td>
<td>15341586</td>
</tr>
<tr>
<td>gi</td>
<td>15226724</td>
</tr>
<tr>
<td>gi</td>
<td>8177711</td>
</tr>
<tr>
<td>gi</td>
<td>4680197</td>
</tr>
</tbody>
</table>
Output - alignments

>gi|7439987|pir||T04346 glycine-rich RNA-binding protein - rice
>gi|2624326|emb|CAA05728.1| OsGRP1 [Oryza sativa (japonica cultivar-group)]
  Length = 160
  Score = 55.5 bits (132), Expect = 5e-07
  Identities = 26/65 (40%), Positives = 42/65 (64%), Gaps = 1/65 (1%)
  Frame = -3

Query: 45840 QHKVYVGNLPWFTQPDLRNHSKFGTIVSTRVLHDKRGRNVFAFLSFTSGEE-RDAA 45664
  +++ +VG L W T    L   FS FG I+ ++++DR+TGR+R F F++F+S + RDA
Sbjct: 7     EYRCFVGLAWATDRSAAFSTFGEILESIINDRETRGSRGFGVFTRFSSEQAMRDAI 66

Query: 45663 LSFNG 45649
  NG
Sbjct: 67    EGMNG 71

>gi|21389168|gb|AAM50518.1|AF510212_1 nam-like protein 15 [Petunia x hybrida]
  Length = 306
  Score = 54.7 bits (130), Expect = 8e-07
  Identities = 34/86 (39%), Positives = 45/86 (51%), Gaps = 4/86 (4%)
  Frame = +2

Query: 4028 WYFFSRRENN--KGNRQSRTTVSGKWKLGTGESVEVKDWGFCSEGFRGKI--GHKRVSFV 4195
  WYFF+ R+     G R +R T +G WK TG+ E+     F G + G K+ LVF
Sbjct: 40    WYFFNLDRDKRYTGLRTNREATEAGYWKTGTGKDEI----------FHGGVLVMKKTLLV 89

Query: 4196 LDGRYPDKTSDKDWHIHEFHYDLLPEH 4273
  GR P K++WV+HE Y L EH
Sbjct: 90    YRGRAPKGQKTNWVMHE--YRLETEH 113
### Output - statistics

Database: /scratch/data/nr.10  
Posted date: May 28, 2003 10:42 AM  
Number of letters in database: 999,354  
Number of sequences in database: 3141  

<table>
<thead>
<tr>
<th>Lambda</th>
<th>K</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.318</td>
<td>0.135</td>
<td>0.401</td>
</tr>
</tbody>
</table>

Gapped  
<table>
<thead>
<tr>
<th>Lambda</th>
<th>K</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.267</td>
<td>0.0410</td>
<td>0.140</td>
</tr>
</tbody>
</table>

Matrix: BLOSUM62  
Gap Penalties: Existence: 11, Extension: 1  
Number of Hits to DB: 158,501,972  
Number of Sequences: 3141  
Number of extensions: 3265826  
Number of successful extensions: 7608  
Number of sequences better than 10.0: 80  
Number of HSP's better than 10.0 without gapping: 2237  
Number of HSP's successfully gapped in prelim test: 0  
Number of HSP's that attempted gapping in prelim test: 0  
Number of HSP's gapped (non-prelim): 7574  
length of database: 999,354  
effective HSP length: 116  
effective length of database: 634,998  
effective search space used: 21092093568  
frameshift window, decay const: 50, 0.1  
T: 12  
A: 40  
X1: 16 (7.3 bits)  
X2: 38 (14.6 bits)  
X3: 64 (24.7 bits)  
S1: 41 (21.7 bits)
Karlin Altschul statistics

- Bit score
  \[ S' = (\lambda S - \ln K) / \ln 2 \]

- E-value
  \[ E = K m n e^{-\lambda S} \]
  - \( S \) is the “raw score”
  - \( \lambda, K \) depend on scoring system (matrix) used
  - \( m \) – effective query size
  - \( n \) – effective database size
  - \( mn \) – effective search space
  - Effective sizes take into account “edge effects” and are smaller than the actual sizes
Choosing effective database size

• To get consistent results, must specify the effective searchspace size
• Calculated by

\[
\text{Boolean } \text{BlastCalculateEffectiveLengths} (\text{BLAST_OptionsBlkPtr } options, \text{Int4 dbseq_num, Int8 dblen, Int4 length, BLAST_KarlinBlkPtr } kbp, \text{Int4Ptr effective_query_length, Int4Ptr length_adjustment})
\]

• This can be accomplished by setting effective database size appropriately when breaking up the database
• Can choose an arbitrary (reasonable) value for effective database length and use it
Some programs which use Methods 1 & 2

- TurboBLAST® (TurboWorx)
  - Distributes databases
  - Integrated job distribution
    - Does “smart” scheduling based on available memory
- Blackstone PowerBLAST
- Platform blastparallel scripts
- mpiBlast (LANL)
  - Uses mpi
- (Pseudo parallel blast) pp-blast
  - Uses pvm
- EPFL Blast
  - Uses ad hoc job distribution
Method 3 – Break up a long query

- Effectively searching parts of the original query in parallel
- Same as Method 1, except must decide where to break query
  - Effect of query length on performance
    - Short queries don’t amortize startup cost
    - Long queries cause to course grained granularity
  - Want to break query at places where there are no hits
  - Must account for effect of query size on statistics
Effect of query size on performance

Query performance

Query performance (n/sec)

Query length

run1
run2
Picking a “split point”
Restrict search to a portion of the database

- Example: genomescan
  - Mask repetitive elements with Repeatmasker
  - Perform initial identification of genes using Genscan
  - Blastp is used to search nr for predicted peptides, results which exceed a specified significance are retained
  - Blastx used to compare masked genomic sequence against retained peptides with increased gap penalties and relaxed EW-value cutoff
  - Genomescan uses blastx results to refine genes

- Example: use Repeatmasker to identify regions which are unlikely to contain genes
Use blast to figure out where genes are

- Use overlapping blast searches

Query 1

Query 2

Why isn’t this in query 2?
Sum Statistics

>gi|125207|sp|P27791|KAPA_RAT cAMP-dependent protein kinase, alpha-catalytic subunit (PKA C-alpha)

Score = 35.0 bits (79), Expect = 0.66
Identities = 14/44 (31%), Positives = 28/44 (62%)
Frame = -1

Query: 65867 QVLSDKGYDGAADVWSCGVILFVLMAAGYLFDEPNLMTLYKRV 65736
+++ KGY+ A D W+ GV+++ + AGY PF + +Y+++ 
Sbjct: 209 EIILSKGYN-KAVDWALGVLIYMAAGYPPFFADQPIQYEKI 251

Score = 32.3 bits (72), Expect(2) = 0.007
Identities = 15/51 (29%), Positives = 25/51 (48%), Gaps = 6/51 (11%)
Frame = -3

Query: 66396 XXXXXXXXXXXXCSL------LQPENLILDANGVLKVSDFGLSAFSRQVR 66262
SL L+PENL++D G ++V+DFG F+++V+ 
Sbjct: 146 FYAAQIVLTFEYLHSDLILYRDNLKPNLLIDQQYIQVTDFF----FAKRVK 193

Score = 28.1 bits (61), Expect(2) = 0.007
Identities = 10/18 (55%), Positives = 13/18 (71%)
Frame = -2

Query: 66154 GLLHTACGTPNYVAPEVL 66101
G T CGTP Y+APE++ 
Sbjct: 194 GRTWTLCGTPYLAPEII 211
Picking size of overlap

- Search overlapped area twice – want it to be small
- Want separation between overlaps to be large
- Want overlap large enough to contain linked HSPs
  - Set blast parameters to limit gap allowed between HSPs (maximum intron size) in sum statistics
- Use a multistage search
Summary

- Get yourself a cluster
- Break up blast jobs and distribute them on the cluster
  - Method 1 – split up input at sequence boundaries
  - Method 2 – split up database at sequence boundaries
  - Method 3 – split up a single sequence query