1 What's New In This Release?

I frequently find myself wanting to make a quick comparison of two long sequences, such as mammalian chromosomes. Even with fairly distant comparisons, such as human vs. chicken, it is often adequate to do this at much less sensitivity than is provided by the Blastz defaults. The command

```
blastz human.chr4 chicken.chr4 C=3 T=2 Z=10 > hsa4.gal4.bz
```

runs in about 5 minutes (and in about 400 Mb of RAM) on my 1Ghz workstation. (Human chr4 is 191 megabases long and chicken chr4 is 90 megabases.) Setting `C=3` avoids the computation of gapped alignments, and using `T=2` instead of the default `T=1`, lowers execution time by a factor of 5-10. Raising Z above the default of 1 increases speed still more, and also helps somewhat with the space consumption. For two similar chromosomes, you may want to use 14-bp matches as seeds, though your machine should have 2Gb of RAM to make this work well. The command

```
blastz human.chr chimp.chr1 C=3 T=4 Z=50 > hsa1.pan1.bz
```

runs about two minutes (and 1.5Gb) on my machine. (Each chromosome is about 250 megabases.)

2 Introduction

This directory contains source code for the Blastz alignment program, which is used by PipMaker and for the whole-genome alignments available from the UCSC Browser. The other code used by PipMaker, e.g., for producing PDF files containing percent identity plots, can be downloaded separately from http://bio.cse.psu.edu/, though be warned that it uses an earlier version of Blastz.

Blastz output can be browsed with the LAJ interactive alignment viewer (also available at http://bio.cse.psu.edu/), converted to traditional text-style alignments using the LAT program that comes with LAJ, or parsed by programs that you write. In this directory, and in the set of PipTools available at http://bio.cse.psu.edu/, there are several examples of programs that process Blastz output.

I make three requests to users of this source code. First, if the code contributes in some significant way to a manuscript that you publish, please acknowledge that fact in the paper and cite the Blastz paper in Genome Research (January 2003). Second, please help me prepare useful documentation, as follows. The current documentation is brief by design; I am wary of writing too much and thereby discouraging users from reading any of it. If you have any questions that you can't answer in a few minutes, or if you have suggestions for either the code or this documentation, please write me at webb@bx.psu.edu so I can learn what users really want. Third, if you using this code for comparative evaluation of another program, please check with me that you have Blastz configured in a reasonable way for the data you are using. (If you want to see why this concerns me, look at: http://biomedcentral.com/1471-2105/5/73.)

We use Blastz in two modes, PipMaker (where the typical job is to compare orthologous BAC-sized sequences) and for whole-genome alignments (where the typical job is to compare two 10-Mb sequences that aren't orthologous). Optimal configuration for Blastz differs between these two uses. See below.

3 Blastz Command-Line Options

Typing just "blastz" (or whatever you name it) should get a "usage" usage message like the following.

```
blastz.v7 command options (e.g., "blastz seq1 seq2 B=0 C=2"):
Default values are given in parentheses.
  m(80M) bytes of space for trace-back information
  v(0) 0: quiet; 1: verbose progress reports to stderr
```

```
B(2) 0: single strand; >0: both strands
C(0) 0: no chaining; 1: just output chain; 2: chain and extend;
     3: just output HSPs
E(30) gap-extension penalty.
G(0) diagonal chaining penalty.
H(0) interpolate between alignments at threshold K = argument.
K(3000) threshold for MSPs
L(K) threshold for gapped alignments
M(0) mask any base in seq1 hit this many times; 0 = no dynamic masking
O(400) gap-open penalty.
P(1) 0: entropy not used; 1: entropy used; >1 entropy with feedback.
Q load the scoring matrix from a file.
R(0) antidiagonal chaining penalty.
T(1) 0: W-bp words;  1: 12of19;  2: 12of19 without transitions.
                     3: 14of22;  4: 14of22 without transitions.
W(8) word size (unused unless T=0)
X(10*(A-to-A match score)) X-drop parameter for ungapped extension.
Y(O+300E) X-drop parameter for gapped extension.
Z(1) increment between successive words in sequence 1.
```

These options are used in commands such as

```
blastz sequence1 sequence2 B=0 C=2 K=2200 > blastz.out
```

or:

```
blastz sequence1 sequence2 C=3 T=2 Z=10 > blastz.out
```

The first command searches only one direction (strand) since B=0, requires matches to be in the same order in both sequences since C=2, and lowers the score threshold (relative to the substitution scores listed at the top of Blastz's output) from 3000 to 2200. (Incidentally, values of K much below 2000 rarely make sense.) The second command computes only gap-free alignments using relatively few computer resources (though at lower sensitivity).

These command-line options account for most of my normal use of blastz. For genome-scale alignments, we frequently set a relatively high initial threshold on the score for gapped alignments (e.g., `L=10000` for human/chicken), but recapture low-scoring alignments that are within a few tens of kb of high-scoring alignments by using `H=2200`, which performs a high-sensitivity search between each two adjacent "anchoring" alignments.

In summary, for high-sensitivity alignment of genomes where the second is in long contigs, try `H=2200`, or if the second is in individual reads try `K=2400`. To search just one strand, `B=0`. To require matches to be in the same order in both sequences, `C=2`. For alignments of really long sequence (but at a modest cost in sensitivity), try `T=2` for distant vertebrates or `T=4` for close species. For really quick peeks at long sequences use `C=3 T=2` or `C=3 T=4` (aligns without gaps). For an approximately a k-fold further speed-up in the computational phase that scans the second sequence, add `Z=k`. For comparing sequences closer than human-mouse, e.g. two primates, I restrict gaps in alignments to at most 100 bp using `Y=3400`, so that new interspersed repeats elements break the local alignment, rather than smearing it (and also speeding up Blastz). I recommend ignoring all parameters not mentioned in this brief document. If you want to align human and mouse sequences, there are other useful tools in this directory; see below.

4 Blastz Output

The first line, `#:lav`, is used to tell if you're looking at the right kind of file. Then comes a "`d {`" stanza that records the substitution scores and the main command-line option values, followed

by a "`s {`" stanza giving the names of the sequence files and the numbers of basepairs, and a "`h {`" stanza giving the FastA headers. We've changed the output format so that nothing is printed for contigs in the second sequence that produce no alignments. If you need to know the contig number and orientation, you can find them in the "`s {`" stanza's entry for the second sequence. For instance, "`"mouse" 1 10000 0 3`" refers to the file named "mouse", 0 for forward direction (1 for reverse complement) of contig number 3 (starting with 1), positions 1-10000. At the very end of the output, there are two little stanzas, "`x {`" and "`m {`" that tell how many regions have been dynamically masked and what those regions are.

An actual alignment looks like this:

```
a {
  s 41249
  b 27347 5196
  e 27406 5259
  l 27347 5196 27365 5214 58
  l 27368 5215 27385 5232 72
  l 27386 5239 27406 5259 67
}
```

This gives the score, beginning positions in the two sequences, ending positions, then a list of the successive gap-free pieces, each specified as begin1, begin2, end1, end2, pct-ident. For reverse complements of the second sequence, alignment positions are given for the flipped sequence, not the original one.

5 Configuring Blastz

If you want to use Blastz for comparing two large genomes, I recommend using the default `T=1` option for genomes at the distance of mammal-vs-marsupial, or the much faster but somewhat less sensitive T=2, and cutting the genomes into 10-Mb pieces that overlap by, say, 10 kb. For human-mouse genome comparisons were using 10-Mb pieces of human and 30-Mb pieces of mouse. For human-human we use just 10-Mb pieces, because then Blastz can easily detect when a piece is being aligned to itself and correctly handle the trivial self-alignment. For more sensitive alignments, you can try K=2500. For really sensitive alignments of short (under 1Mb) sequences, say between a mammal and a fish, you could try T=0 W=6 K=2200 (seeds with 6-bp exact matches).

An earlier version of Blastz used a linear-space algorithm for dynamic programming. We've gone back to a simpler system that can use more memory. If Blastz complains that it ran out of trace-back memory, you can adjust the `m` parameter. If it runs out of trace-back memory, then Blastz splits a long alignment into several shorter ones.

For best performance under some circumstances, you can consider use of the C-compiler flag `-DUSE_OBSTACK` (see the Makefile). It helps with performance in general, and is necessary to work around a bug in some versions of gnu malloc on large memory Linux systems (e.g. the version of redhat 6.2, with 1GB). Obstack.c is in glibc, but it also comes with many gnu programs (like gcc and emacs), so it's likely to already be on your computer if you are running something other than Linux.

6 Human-Mouse Alignments

For the full-genome alignments, we're using a strategy proposed several years ago by Arian Smit: we apply Blastz to sequences from which the lineage-specific interspersed repeats have been removed, then restore the coordinates in the original sequences. This directory includes four programs that perform these operations, and the Makefile illustrates their use.

7 Mouse-Rat Alignments

Currently, the RepeatMasker library of rodent interspersed repeats is substantially less complete than the human library, so mouse and rat can't be aligned to each other at the default Blastz sensitivity without producing a huge number of spurious alignments. As a temporary work-around, I aligned with the options `T=2` (which has the nice side-effect of making Blastz run about 10 times faster in this case), `L=50000` (which throws away all alignments scoring less than 50000, which is the best possible score for a 500-bp alignment), and `H=2500` (which searches around and between these very high scoring alignments for low-scoring but still significant ones).