darwin: a Scalable Version Control System for Genomic Data

Danny McClanahan, Vanderbilt University Software

Abstract

- Synthetic biologists create genomes by editing DNA text directly.
- Changes made are difficult to track, which leads to security problems.
- No software exists to track changes which works with genome-scale data.
- darwin is a software package to document and track collaborative changes to DNA on the genome scale.

Basic Biology Review

- ORF (open reading frame) codes for a protein
- Are therefore the interesting parts of a gene
- Can have multiple ORFs per gene
- Translated by ribosomes in the cell
- Has special start and end markers
  - Ribosome uses these to determine where to begin and end translation into proteins

What is Version Control?

- Record every change made to a file or set of files
- When, What, Who
- Merge changes by multiple collaborators
- Ensures every member of team has updated copy
- Typical tool used is called **git**

How Git Processes Files

- git is a line-based system
- Only records lines added and deleted

AAAAAAAAA	AAAAAAAAA	
BBBBBBBBB	CCCCCCCC	-BBBBBBBBB
CCCCCCCC	DDDDDDDD	+DDDDDDDD
previous	**current**	**changes recorded**

- The more lines in a file, the longer it takes git to process.
- This makes it inefficient for processing DNA files

What darwin Does

- darwin **preprocesses** DNA files before putting them through git
- Create temporary file which is optimized so git performs **fewer operations** and **runs faster**
- Put temporary file in git
- Reconstruct original file from temporary file
- Makes version control of genomic data feasible by increasing the speed at which git processes data.

Approach Part 1: **Split** by ORF

- FASTA/GenBank/ApE/etc typically formatted in fixed-length lines
- e.g.:
  - FASTA (typically 50 or 70 characters per line):

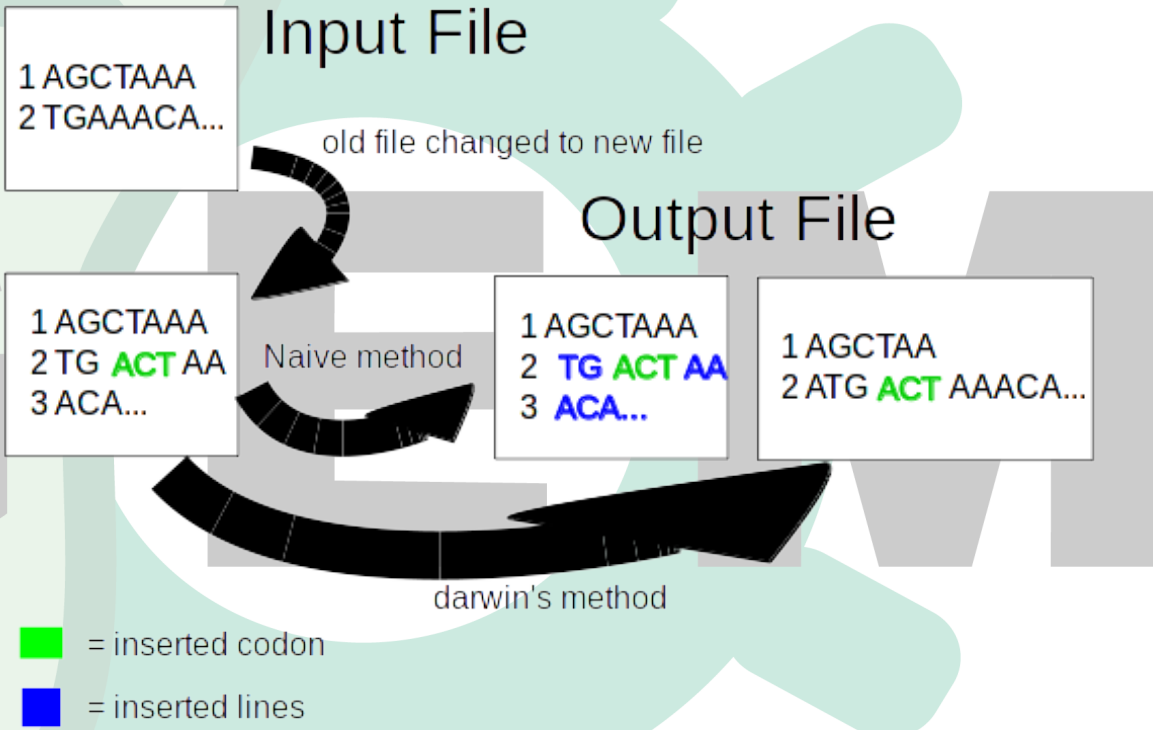
```
CATACAATCCAGGTTTTAATCATCAGAAATCACAGTCCTATTGTCTTCTGCACAGACCCAAACACACTTG
GAGGTCATGTTCAATATGAATACCTCACAGAGAAGGAAATTTACACGCGAGAAGTACATCTGCAGAAAGC
CAGCTGGCATGTCAACCATTCAAAAACTCAGGGTGTTCTGGATAAAGAAGACTCAGGAAGACAAGTATGA
AGCATAATCTGTGACATTCCATGCGGCAGACATTAGACACATACAAGAGAGTTGTTGGAAAGCGGAATTT
ATCTTCATATAAACAACACTGAGCTAAATCTCAATATTTCAGATCTCTAGAACTATCCATCAGTGAAATG
```

  - ApE (typically 76 characters per line):

```
   1 TCGCGCGTTT CGGTGATGAC GGTGAAAACC TCTGACACAT GCAGCTCCCG GAGACGGTCA
  61 CAGCTTGTCT GTAAGCGGAT GCCGGGAGCA GACAAGCCCG TCAGGGCGCG TCAGCGGGTG
 121 TTGGCGGGTG TCGGGGCTGG CTTAACTATG CGGCATCAGA GCAGATTGTA CTGAGAGTGC
 181 ACCATATGCG GTGTGAAATA CCGCACAGAT GCGTAAGGAG AAAATACCGC ATCAGGCGCC
```

Approach Part 1: **Split** by ORF

- Remove formatting of FASTA/ApE/ GenBank/etc
- Split file into lines by ORF
- Changes to single ORF now only affect single line
- Temporary file produced is now much smaller



Approach Part 1: **Split** by ORF

- Output files now look like this

CATACAATCCAGGTTTTAATCATCAGAAATCACAGTCCTATTGTCTTCTGCACAGACCCAAACACACTTG
GAGGTC
ATGTTCAATATGAATACCTCACAGAGAAGGAAATTTACACGCGAGAAGTACATCTGCAGAAAGC
CAGCTGGCATGTCAACCATTCAAAAACTCAGGGTGTTCTGGATAA
AGAAGACTCAGGAAGACAAGT
ATGA
AGCATAATCTGTGACATTCCATGCGGCAGACATTAGACACATACAAGAGAGTTGTTGGAAAGCGGAATTT
ATCTTCATATAA
ACAACACTGAGCTAAATCTCAATATTTCAGATCTCTAGAACTATCCATCAGTGAA
ATG

- Note that lines are now varying length, and alternating between ORF and non-ORF
- Adding or modifying an ORF now only changes a single line of output

Approach Part 2: Edits **within** ORF

- Consider adding a few amino acids at the beginning of a long ORF:
  - Before: ATGAGAGGCGGTTGC...
  - After: ATG**AAAAGCATA**AGAGGCGGTTGC...
- Since git only sees changes in lines, it counts the same as adding and removing an **entire ORF**
- This could be **thousands** of characters changed for a single small insertion

Approach Part 2: Edits **within** ORF

ATGAGAG…
previous

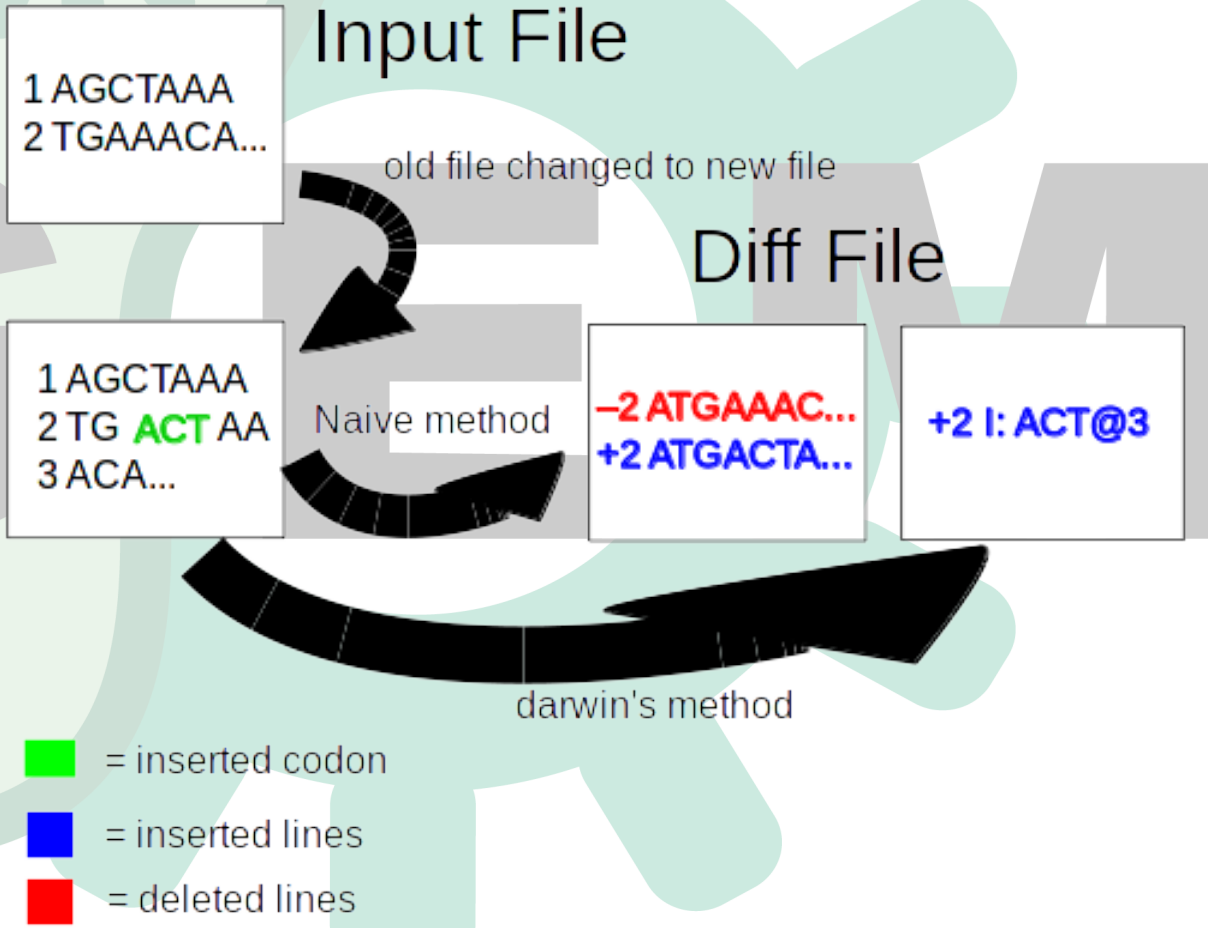
ATG**AAAAGCATA**AGAG…
current

-ATGAGAG…
+ATGAAAAGCATA**AGAG…**

changes recorded

Approach Part 2: Edits **within** ORF

- Identify ORFs that have only small edits between two versions of file
- Find only those small changes that were made and record those
- Actual ORF can be reconstructed from previous ORF + changes



Approach Part 2: Edits **within** ORF

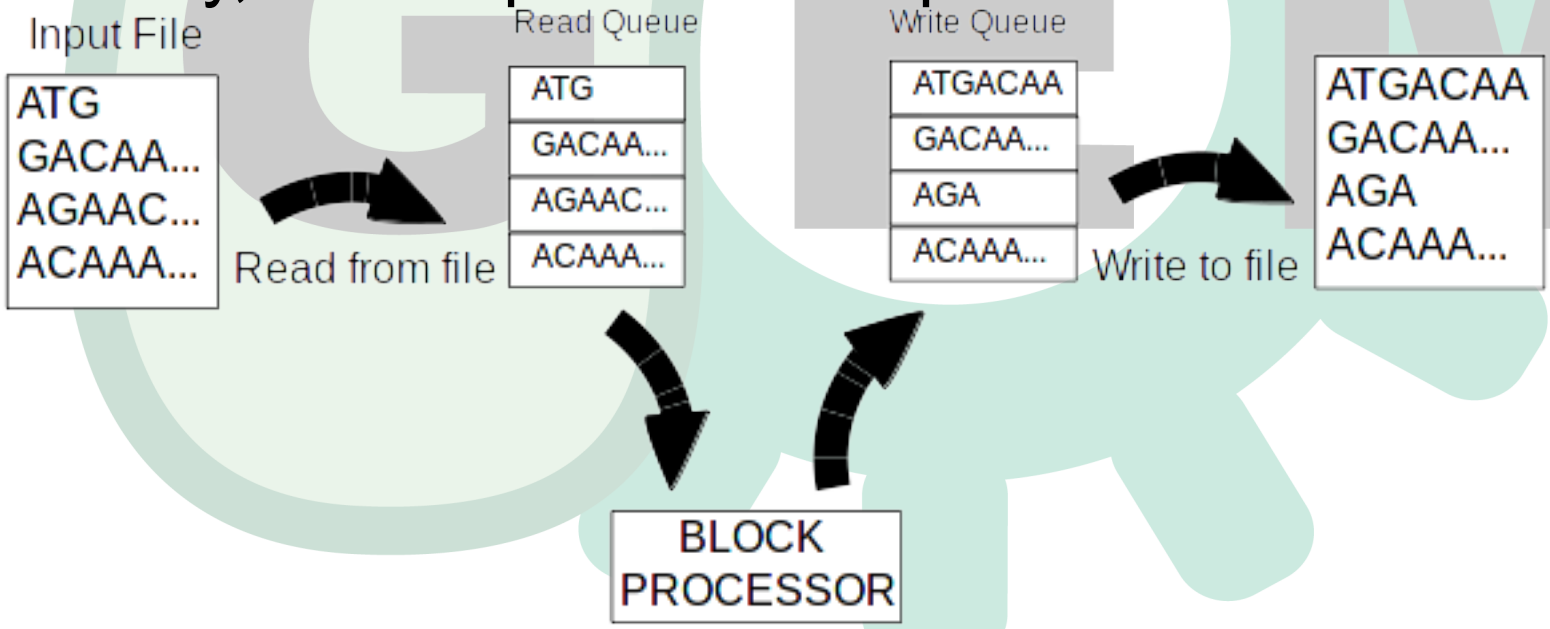
- Previous example:
  - Before: ATGAGAGGCGGTTGC...
  - After: ATG**AAAAGCATA**AGAGGCGGTTGC...
- This turns into:
  - ATGAGAGGCGGTTGCA...
  - **+AAAAGCATA@3**
- Short line of edits added, not entire long ORF

Approach Part 3: Use of **Concurrency**

- Water bucket analogy
- File I/O (input-output) is extremely slow
- darwin has to do both input and output
- So use concurrency to continue to do work while waiting for slow file operations

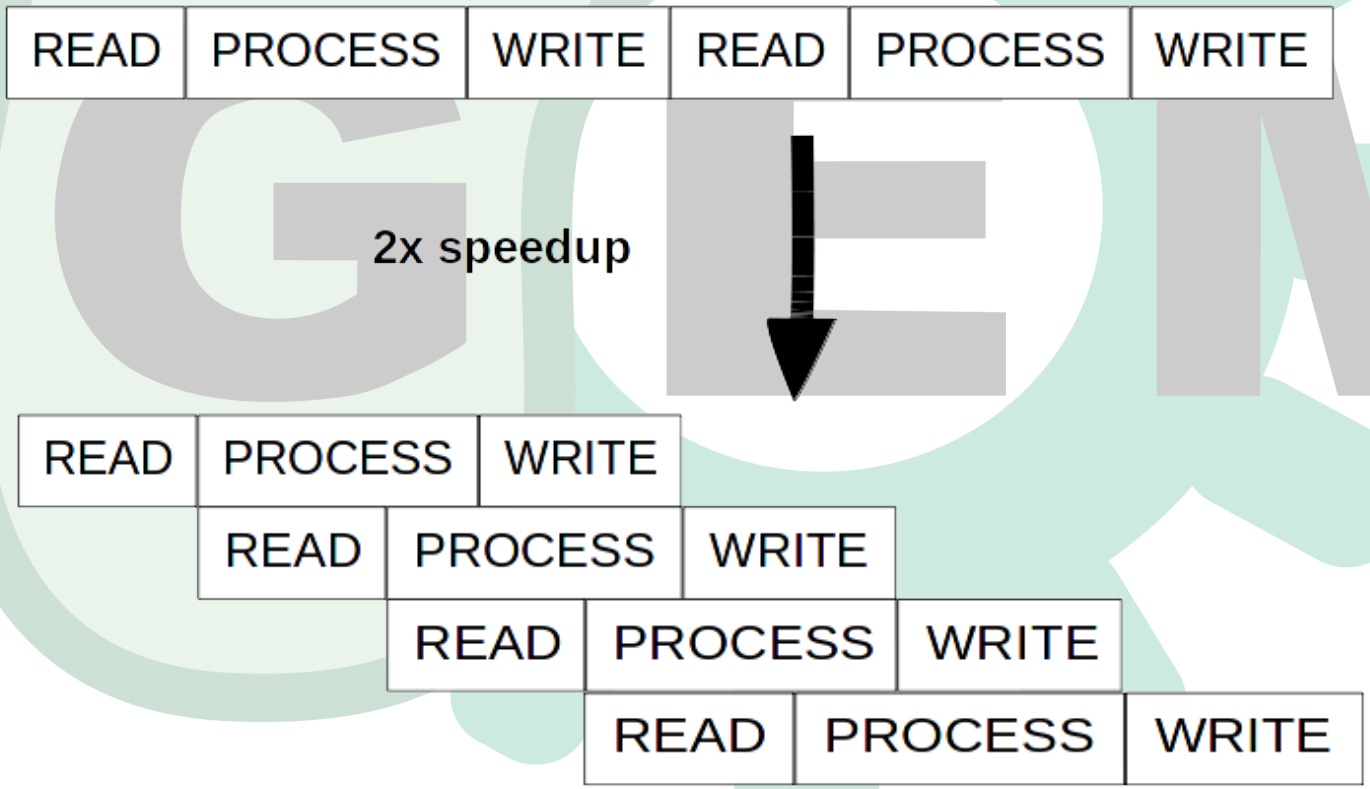
Approach Part 3: Use of **Concurrency**

- Create queues of “buckets” of input and output
- First bucket passed from file reader to processor
- File reader continues reading while processor completes
- Finally, bucket passed from processor to writer



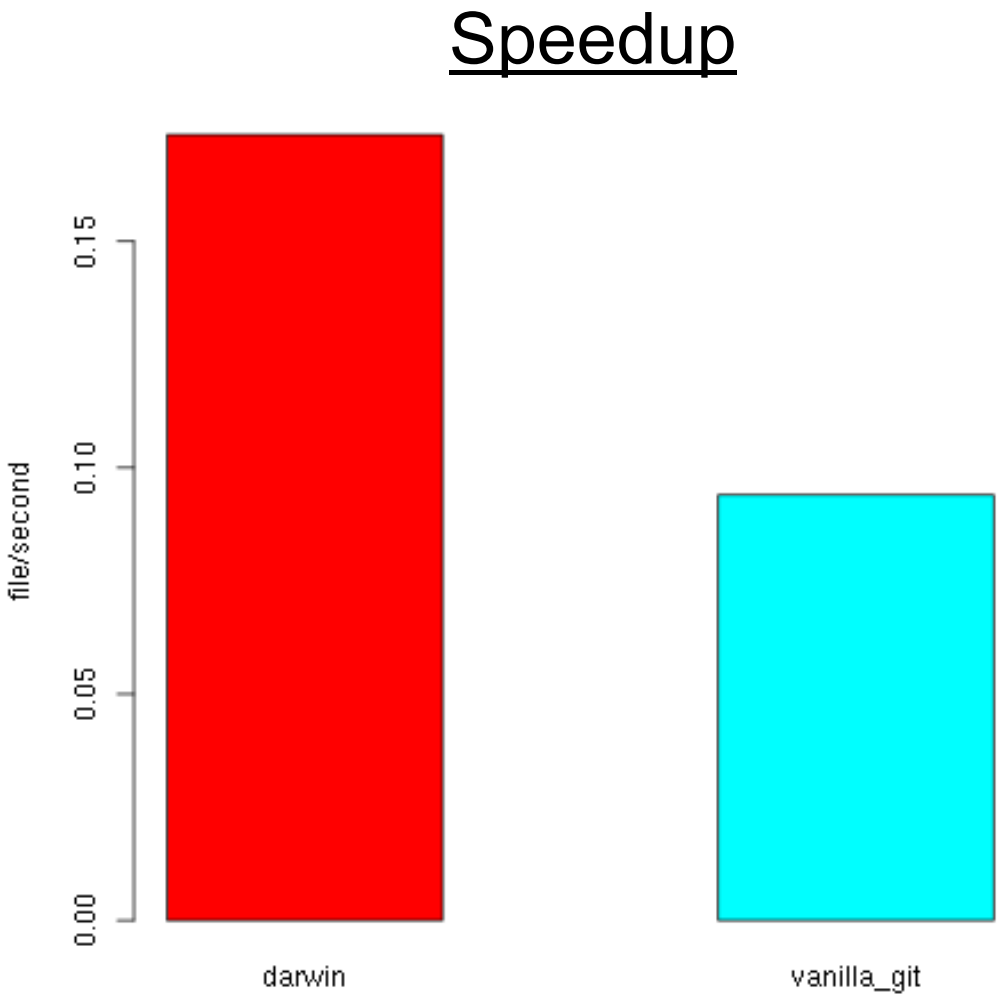
Approach Part 3: Use of **Concurrency**

- Perform four cycles side-by-side in same time as two cycles without concurrency
- Massive pipelined speedup available



Results

- Tested on multiple iterations of ApE files from Vanderbilt wetware team
- darwin made processing files with git about twice as fast



Results

- Data about experimental setup
  - 40,000 trials run on four successive iterations of a real-life DNA file
  - “wall-clock time” used to measure time actually visible to the user
- Why do results matter?
  - This experiment shows that even a draft copy of the software can achieve extremely impressive results.

Future Work

- More filetypes:
  - 2bit, SAM/BAM, etc
- GUI
- Further optimization

Project Summary

- darwin is a software package to document changes to DNA.
- Allows for easy, standardized, and collaborative editing on DNA data up to the genome scale.
- Builds off of tested and proven version control software.
- Uses algorithms to preprocess DNA files and log changes twice as fast as the current method.

Acknowledgements

- Mitchell Gordon, for software development.
- Jules White, for advice and help.
- VUSE, and specifically the EECE department, for their support throughout this project.