A LAYERED GENOTYPING-BY-SEQUENCING PIPELINE USING GALAXY

Rudiger Brauning, Simon Guest, Alan McCulloch, Russell Smithies
AgResearch, New Zealand

Galaxy Community Conference 2013, Oslo
WHAT’S THIS ALL ABOUT?

• A *layered* approach to *data parallelism*
  • Execute the *same* command in parallel on *different* pieces of the data
  • Layered => hide the details of doing this from the user

• Greatly accelerates processing on compute farm

Layer Analogy from Networks

<table>
<thead>
<tr>
<th>TCP</th>
<th>IP</th>
<th>Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seamless transport</td>
<td>Whole process view</td>
<td>Node-to-node transfer</td>
</tr>
<tr>
<td>Split, schedule, recombine</td>
<td>Process one fragment</td>
<td>Galaxy Workflow</td>
</tr>
<tr>
<td>bwa/blast/...</td>
<td>Data Parallel Driver</td>
<td>Bioinformatics pipeline</td>
</tr>
</tbody>
</table>
OUR EXAMPLE IMPLEMENTATION ON GALAXY

• We wrote a Python script called *tardis*
  • splits input files, schedules parallel jobs, recombines output
  • plumbed into Galaxy via Cheetah templates in tool XML files
  • user-switchable via checkbox on tool webpage

• Do *something* to a \{fasta, fastq, sam, bam\} file in parallel
  • For your choice of *something* 
  • extensible, more to come
Map with BWA for Illumina (version 1.2.3)

Will you select a reference genome from your history or use a built-in index?:
Use a built-in index

Select a reference genome:
Sheep OAR31: sheep.v3.0.14th.lowercase.final.fa

Is this library mate-paired?:
Single-end

FASTQ file:

FASTQ with either Sanger-scaled quality values (fastqsanger) or Illumina-scaled quality values (fastqillumina)

BWA settings to use:
Commonly Used
For most mapping needs use Commonly Used settings. If you want full control use Full Parameter List

Suppress the header in the output SAM file:

Use HPC cluster:

the input file will be split, and chunks will be processed on the compute farm. The chunk outputs will be pasted together before entry into your history

Chunk size:
1500000
the number of sequences each split should contain

Sample rate (leave empty to process the complete file):
for example entering .001 will randomly sample and process roughly 1 in every 1000 records
GENOTYPING BY SEQUENCING (GBS)

Already a *data parallel* process

If you’re happy to process each barcode as a unit, simply run each of these processes on your compute farm.
WHOLE GENOME SHOTGUN SEQUENCING (WGS)

No *data parallelism*

Even on our biggest server, this pipeline can take a loooong time …

(a week or more)

… unless we can
• split the input file
• process in parallel

HP DL585 G7
64 cores
512 GB RAM
GENERAL CASE

$tardis$ splits, manages cluster jobs, and recombines ... ... transparently
AGRESEARCH COMPUTE FARM

48 x HP BL 485c, 4 core, 16 GB RAM

4 x HP DL585 G7, 48/64 core, 512 GB RAM

2 x HP DL385 G7 Fileservers, 16 core, 64 GB
200 TB storage (ZFS backed NFS over 10Gb Eth)

Key enabler: *uniform environment*

i.e. all compute servers
  • run identical software
  • see same network filesystem
KEY TECHNOLOGIES

AgResearch Bioinformatics RPM Repository

http://rpm.agresearch.co.nz
Map with BWA for Illumina (version 1.2.3)

Will you select a reference genome from your history or use a built-in index?:
- Use a built-in index

Select a reference genome:
- Sheep 0AR31:sheep.v3.0.14th.lowercase.final.fa

Is this library mate-paired?:
- Single-end

FASTQ file:
- Upload the FASTQ file to use.

FASTQ with either Sanger-scaled quality values (fastqsanger) or Illumina-scaled quality values (fastqillumina)

BWA settings to use:
- Commonly Used
  - For most mapping needs use Commonly Used settings. If you want full control use Full Parameter List

Suppress the header in the output SAM file:
- Check this box to suppress the header in the output SAM file.

BWA produces SAM with several lines of header information

Use HPC cluster:
- Check this box to use the HPC cluster.
  - The input file will be split, and chunks will be processed on the compute farm. The chunk outputs will be pasted together before entry into your history

Chunk size:
- Enter the chunk size.
  - The number of sequences each split should contain

Sample rate (leave empty to process the complete file):
- Enter the sample rate.
  - For example entering .001 will randomly sample and process roughly 1 in every 1000 records
KNOWLEDGE FOR TOOL-WRAPPERS

$ tardis.py -h

tardis is a script which "conditions" a command for execution on a cluster, 
launches the conditioned command as a series of jobs, and collects and collates the output.

Usage :

tardis.py [-w] [-c Chunksize] [-s SampleRate] [-d workingRootPath] [-k] [-v] 
   any comand (with optional conditioning directives)

   -w : Run as part of a workflow. After launching all of the jobs, tardis waits for all outputs, which is 
      then collated and merged into a single output file, as specified by the output file path. 
      Without this option, the program exits immediately after launching all of the jobs, and output is left 
      un-collated in the scratch folder created by this script.

   -c ChunkSize : when conditioning the input file(s), split into files each containing Chunksize 
      logical records. (A logical record for a sequence file is a complete sequence. For a text file it 
      is a line of text). (If the -s option is used to sample the inputs, the chunksize relates to the 
      full unsampled file. For example if a chunksize of 1,000,000 is specified in combination with a 
      sampling rate of .0001, then each chunks would contain 100 sequences)

   -s SampleRate : rather than process the entire input file(s), a random sample of the records is processed. 
      SampleRate is the probability that a record will be sampled. For example -s .001 will result in 
      roughly 1 in every 1000 logical records being sampled. When the -s option is specified, tardis does not 
      clean up the conditioned input and output - i.e. all of the fragments. These are retained to assist 
      with the Q/C work that is normally associated with a sampled run.

   -d workingRootPath : create the tardis working folder under workingRootPath. If no working root is specified, 
      a default location is used.

   -v : validate the run by doing a dry run. This means that the chunks , scripts and job files etc are all generated but 
      the jobs are not launched. The user can inspect the script and job files to check that their 
      command has been conditioned as envisaged.

   -k : keep the conditioned input and output - i.e. the input and output fragments. Normally these are 
      deleted after the output is succesfully "unconditioned" - i.e. joined back together

   -h : print usage and exit.

see the tardis documentation for more information on the conditioning directives that are supported, and 
examples of conditioned commands.
EXAMPLE INVOCATION

tardis.py -c 1500000 -s 0.0001 -w blastn -query _condition_fastq2fasta_input_/dataset/JHI_High_Low_Sequencing_Data/L01_filtered.fastq.gz -query_gencode 11 -db nt -outfmt 5 -evalue 0.0001 -out _condition_blastxml_output_/dataset/JHI_High_Low_Sequencing_Data/scratch/L01_sample.xml

merge XML output files
INPUT CONDITIONING

• Input conditioning directives are prefixed to filename arguments
• Input files that are prefixed with conditioning directives will
  • If necessary be uncompressed (e.g. fastq.gz to fastq)
  • If necessary be sampled randomly
  • If necessary be format-converted (e.g. fastq to fasta) as implied by the directive
• Split into chunks

_condition_fastq_input_  optionally uncompress and then split
_condition_fasta_input_  optionally uncompress and then split
_condition_fastq2fastq_input_  optionally uncompress and then split into fasta fragments
_condition_pairedfastq_input_  optionally uncompress and then split into pair-end matched fastq fragments, enforcing integrity of pair-end matching by name
_condition_text_input_  optionally uncompress and then split a text file
OUTPUT CONDITIONING

- Output conditioning directives are prefixed to filenames
- Output files that are prefixed with conditioning directives will be
  - Recombined
  - If necessary be compressed

_condition_fastq_output_
_condition_fasta_output_
_condition_fastq2fasta_output

_condition_text_output_ concatenate the output fragments into the prefixed filename and compress.

_condition_sam_output_
_condition_pdf_output_
_condition_blastxml_output_

compress each SAM file to single merged BAM
combine the pdf fragments
combine the blast XML fragments into a single valid blast XML output
#if str($use_hpc) == "yes":
    #if str($use_hpc_samplerate) != "":
        #set $prog = ["tardis.py -w -c ", str($use_hpc_chunksize) , "-s", str($use_hpc_samplerate) , " -d /dataset/galaxy_scratch2011/scratch/tardis python /home/galaxy/galaxy/tools/sr_mapping/ bwa_wrapper.py "]
    #else:
        #set $prog = ["tardis.py -w -c ", str($use_hpc_chunksize) , " -d /dataset/galaxy_scratch2011/scratch/tardis python /home/galaxy/galaxy/tools/sr_mapping/bwa_wrapper.py "]
    #end if
    #set $prog = string.join($prog, " ")
#else
    #set $prog = "bwa_wrapper.py "
#end if

$prog ...
# if str($use_hpc) == "yes":
    ## input file(s)
    #if $paired.sPaired == "paired":
        --input1=_condition_paired_fastq_input_$paired.input1
        --input2=_condition_paired_fastq_input_$paired.input2
    #else:
        --input1=_condition_fastq_input_$paired.input1
    #end if

    ## output file
    --output=_condition_uncompressedsam_output_$output
# else:
    ## input file(s)
    --input1=$paired.input1
    #if $paired.sPaired == "paired":
        --input2=$paired.input2
    #end if

    ## output file
    --output=$output
#end if
WHY AM I TELLING YOU THIS?

• A layered approach to data parallelism may be useful in general

• Our implementation in tardis may be of interest

• We are happy to support adoption of this work by the Galaxy community

Thanks for listening!

simon.guest@agresearch.co.nz
http://bitbucket.org/agr-bifo/tardis