S9350

CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Richard Wilton
Department of Physics and Astronomy
Johns Hopkins University

S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

- A very brief description of short-read alignment
- Arioc: a GPU-accelerated short-read aligner
- What is a “large” genome?
- A software view of a reference genome
- Repetitiveness versus speed
- Performance

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Short-read alignment

Perfect alignment

```
R: CATGTGTGAAGCCTCCATACTTGAGTCCTGAACTGATGAACTAA
           ||||||||||||||||||||||||||||||
Q:         AAGCCTCCATACTTGAGTCCTGAACTGATGAA
```

Alignment with mismatches

```
R: CATGTGTGAAGCCTCCATACCTGAGTCATGAACTGATGAACTAA
           |||||||||||| ||||||| ||||||||||||
Q:         AAGCCTCCATACTTGAGTCCTGAACTGATGAA
```

Alignment with mismatches and gaps

```
R: CATGTGTGAAGCCGCGCGTCCATACATGAGTCATGAAC--ATGAACTAA
           ||||||     |||||| |||||| |||||  |||||
Q:         AAGCCT-----CCATACTTGAGTCCTGAACTGATGAA
```

Scoring example

Parameters	
match	+2
mismatch	−6
gap	−5
space	−3

↓

Scores	
perfect	64
mismatches	48
mismatches and gaps	11

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Short-read alignment

Extract and hash subsequences (“seeds”)

```
Q: AAGCCTCCATACTTGAGTCCTGAACTGATGAA
   AAGCCTCCAT     →  0xDEA5D502
    AGCCTCCATA    →  0x29DEC1F0
     GCCTCCATAC   →  0xDB840577
      CCTCCATACT  →  0x4DBA90D5
       ...
```

Probe hash table to find reference-sequence locations

```
0xDEA5D502:  01:14353363, 01:15536663, 02:06335366 ...
0x29DEC1F0:  01:14353364, 06:20159342, 18:00513566
0xDB840577:  01:14353365, 01:15536665, 05:83754151 ...
0x4DBA90D5:  (none)
```

Look for high-scoring alignments (“extend”) at high-priority reference-sequence locations

```
R: CATGTGTGAAGCCGCCATACCTGAGTCATGAAC--ATGAACTAA
           |||||||||||| |||||| |||||  |||||
Q:         AAGCCTCCATACTTGAGTCCTGAACTGATGAA
```

S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

- A very brief description of short-read alignment
- Arioc: a GPU-accelerated short-read aligner
- What is a “large” genome?
- A software view of a reference genome
- Repetitiveness versus speed
- Performance

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

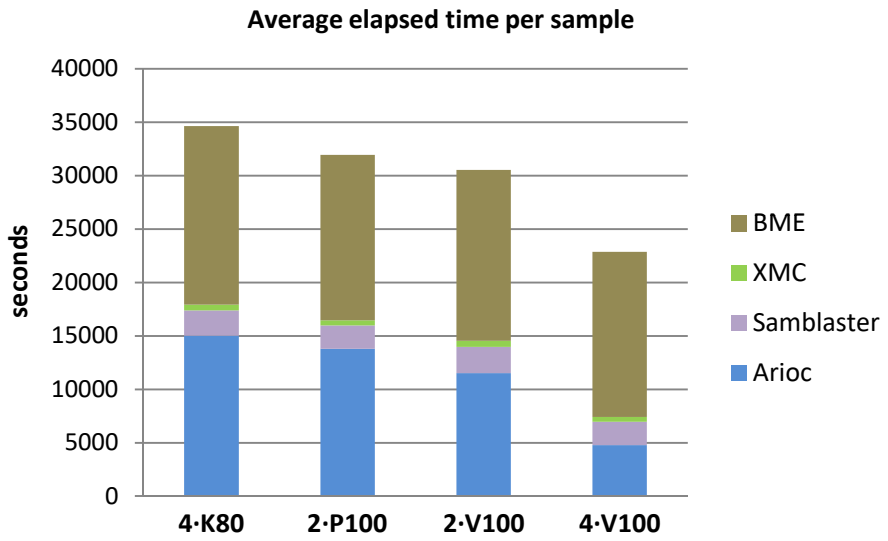
Arioc: a GPU-accelerated short-read aligner

- Speed
  - Short-read alignment is just one step in a processing “pipeline”; the idea is that this step should not be a bottleneck
  - Order-of-magnitude (~10x) faster than CPU-only implementations
- Sensitivity
- Accuracy
- Capable of handling real-world data
  - Full-sized sequencer runs
  - Human reference genome (and larger)

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Arioc is fast

- 1,304 WGBS samples
  - 150bp paired-end
  - Human reference genome
  - Average sample size: 487,757,780 pairs (975,515,560 reads)
- One step in a series of analysis tools
  - Arioc
  - Samblaster
  - Bismark methlylation extractor
- Shared compute nodes at MARCC (Maryland Advanced Research Computing Center)



S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

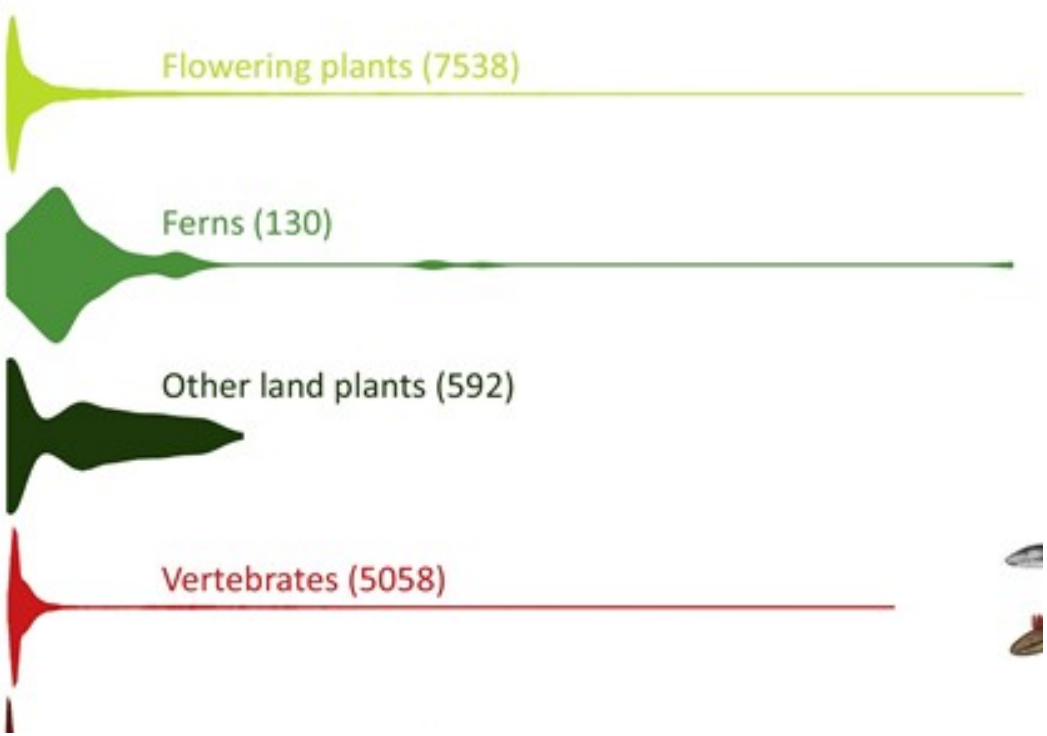
- A very brief description of short-read alignment
- Arioc: a GPU-accelerated short-read aligner
- What is a “large” genome?
- A software view of a reference genome
- Repetitiveness versus speed
- Performance

"Large" compared to what?

- The human genome is a good starting point for comparison
  - About 3 billion nucleotide bases
  - If you number each base position consecutively, you can identify each base with a 32-bit integer!
- Some interesting organisms have genomes that contain much more DNA than does the human genome

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

What is a large genome?



Some large genomes whose DNA has been sequenced

Organism	Size ($\times 10^9$)
Mexican axolotl	32
Pine tree	22
Wheat	14.5
Human	3.2
Mouse	2.7

S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

- A very brief description of short-read alignment
- Arioc: a GPU-accelerated short-read aligner
- What is a “large” genome?
- A software view of a reference genome
- Repetitiveness versus speed
- Performance

Identifying genome locations

- Subunit ID
  - Usually a chromosome number
  - Range of values: 1-127
- DNA strand
  - Forward or reverse complement
  - Range of values: 0-1
- Offset from the start of the DNA sequence
  - Range of values: 0-2,147,483,647

Chromosomes in axolotl genome

Chromosome	Size ($\times 10^9$)
1Q	1.48
2P	1.41
2Q	1.51
3P	1.24
3Q	1.26
4P	1.16
7	2.03
4Q	1.29
8	1.71
5P	1.29
9	1.50
5Q	1.34
10	1.64
6P	1.55
11	1.44
6Q	1.59
12	1.21
13	0.72
14	0.66

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Reference genome position in C++

```
/* 40-bit (5-byte) representation of a J value */
struct Jvalue5
{
    enum bfSize
    {
        bfSize_J =     31,  //  0..30: J (0-based offset into reference sequence)
        bfSize_s =      1,  // 31..31: strand (0: R+; 1: R-)
        bfSize_subId =  7,  // 32..38: subId (e.g., chromosome number)
        bfSize_x =      1   // 39..39: end-of-list flag
    };

    enum bfMaxVal : UINT64
    {
        bfMaxVal_J =     (static_cast<UINT64>(1) << bfSize_J) - 1,
        bfMaxVal_s =     (static_cast<UINT64>(1) << bfSize_s) - 1,
        bfMaxVal_subId = (static_cast<UINT64>(1) << bfSize_subId) - 1,
        bfMaxVal_x =     (static_cast<UINT64>(1) << bfSize_x) - 1
    };

    UINT32  J     : bfSize_J;
    UINT32  s     : bfSize_s;
    UINT8   subId : bfSize_subId;
    UINT8   x     : bfSize_x;
};
```

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Large genome → large lookup tables

Extract and hash subsequences ("seeds")

```
Q: AAGCCTCCATACTTGAGTCCTGAACTGATGAA
   AAGCCTCCAT     →  0xDEA5D502
    AGCCTCCATA    →  0x29DEC1F0
     GCCTCCATAC   →  0xDB840577
      CCTCCATACT  →  0x4DBA90D5
       ...
```

Hash table data-sort sizes

32-bit seeds	# lists to sort	# locations to sort
human	1,263,683,062	3,687,638,902
wheat	2,120,243,009	20,602,998,718

Probe hash table to find reference-sequence locations

```
0xDEA5D502:  01:14353363, 01:15536663, 02:06335366 ...
0x29DEC1F0:  01:14353364, 06:20159342, 18:00513566
0xDB840577:  01:14353365, 01:15536665, 05:83754151 ...
0x4DBA90D5:  (none)
```

Look for high-scoring alignments ("extend") at high-priority reference-sequence locations

```
R: CATGTGTGAAGCCGCCATACCTGAGTCATGAAC--ATGAACTAA
           |||||||||||| |||||| |||||  |||||
Q:         AAGCCTCCATACTTGAGTCCTGAACTGATGAA
```

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

"Sortable" reference genome position in C++

```
/* 64-bit (8-byte) representation of a 40-bit (5-byte) J value */
struct Jvalue8
{
    enum bfSize
    {
        bfSize_J =     31,     //  0..30: J (0-based offset into reference sequence)
        bfSize_s =      1,     // 31..31: strand (0: R+; 1: R-)
        bfSize_subId =  7,     // 32..38: subId (e.g., chromosome number)
        bfSize_x =      1,     // 39..39: flag (used only for sorting and filtering J lists; zero in final J table)
        bfSize_tag =   24      // 40..63: used for sorting (see tuSortJgpu)
    };

    enum bfMaxVal : UINT64
    {
        bfMaxVal_J =     (static_cast<UINT64>(1) << bfSize_J) - 1,
        bfMaxVal_s =     (static_cast<UINT64>(1) << bfSize_s) - 1,
        bfMaxVal_subId = (static_cast<UINT64>(1) << bfSize_subId) - 1,
        bfMaxVal_x =     (static_cast<UINT64>(1) << bfSize_x) - 1,
        bfMaxVal_tag =   (static_cast<UINT64>(1) << bfSize_tag) - 1
    };

    UINT64  J     : bfSize_J;
    UINT64  s     : bfSize_s;
    UINT64  subId : bfSize_subId;
    UINT64  x     : bfSize_x;
    UINT64  tag   : bfSize_tag;
};
```

A bit-packed segmented sort

- The lists are sorted in a call to a CUDA Thrust sort implementation

```
/* Sort the current J-list buffer chunk.  Since each 64-bit value contains a "tag" that associates the value
       with the J list that corresponds to an H (hash key) value, this is in effect a segmented operation. */
thrust::device_ptr<UINT64> ttpJbuf( m_pJbuf->p );
thrust::sort( epCGA, ttpJbuf, ttpJbuf+m_pJbuf->Count );
```

  - The high-order bits identify individual lists so the result is effectively a segmented sort
  - There are more lists than can be uniquely identified in the available high-order bits, so the Thrust sort API is called iteratively

S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

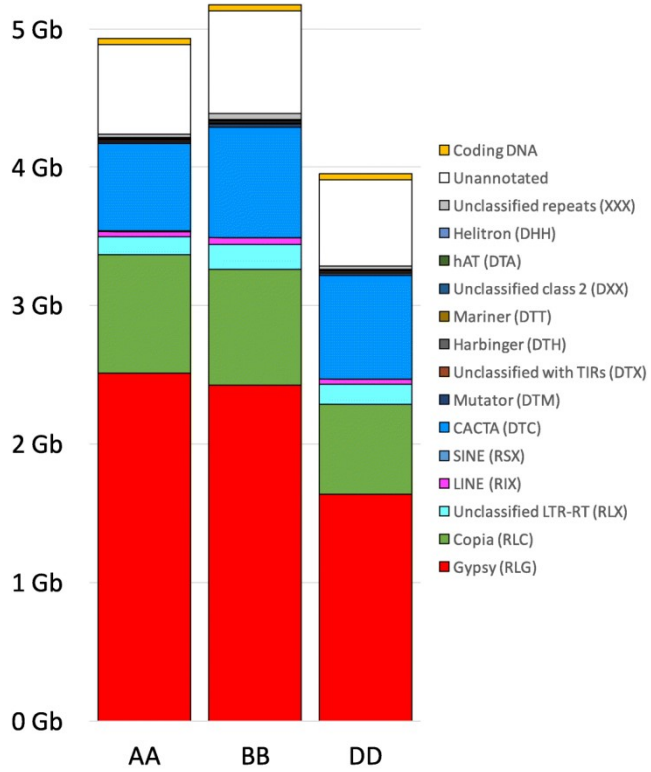
- A very brief description of short-read alignment
- Arioc: a GPU-accelerated short-read aligner
- What is a “large” genome?
- A software view of a reference genome
- Repetitiveness versus speed
- Performance

Large genomes are repetitive

- The genome of complex organisms contains mostly non-coding DNA
- Non-coding DNA in large genomes contains many long and/or repetitive sequences
- The repetitive nature of the DNA affects the way we construct the hash tables (lookup tables)

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

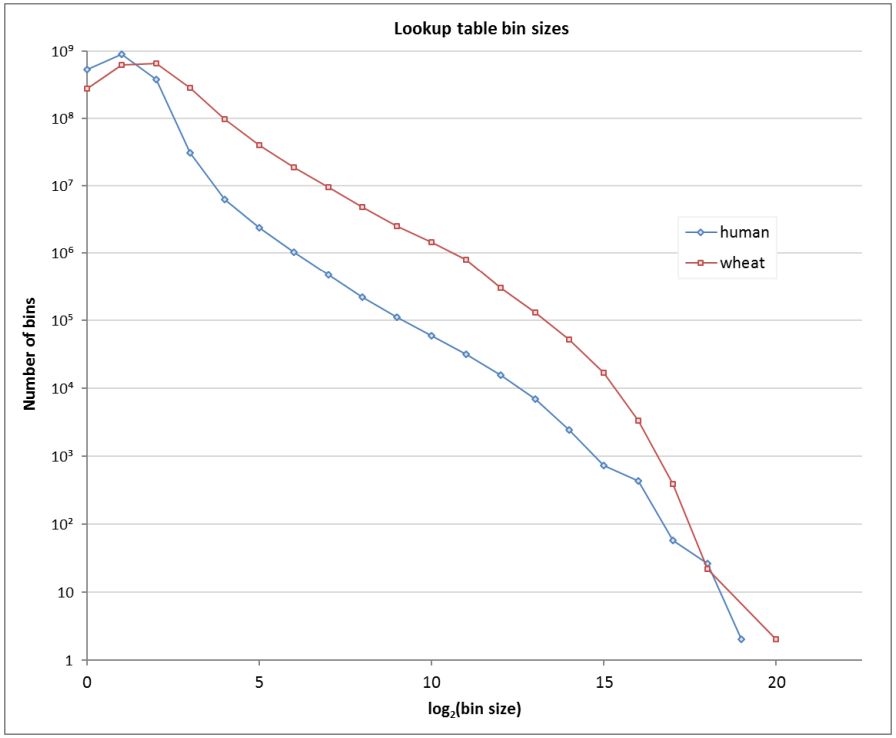
Large genomes are repetitive



- In large genomes, much of the DNA is repetitive:
  - Human: ~50%
  - Bread wheat (Triticum aestivum): ~85%
- Repetitive DNA may contain…
  - Multiple copies of a variety of short subsequences
  - Low-information sequences
    - Homopolymers (e.g., AAAAAAAAAAAA)
    - Tandem repeats (e.g., CGCGCGCGCGCG)
- With a large repetitive genome, any given read may have multiple locations at which it aligns
  - More alignment computations per read
  - Increased post-alignment processing to identify and classify high-scoring mappings for each read

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Repetitive genome → large lookup tables



- Large-genome lookup tables (LUTs) contain more reference-sequence locations per hash value
  - Big LUT bins → more alignments computed
- Large-genome LUTs are hard to optimize
  - Pruning highly-repetitive seed locations decreases sensitivity in the read aligner

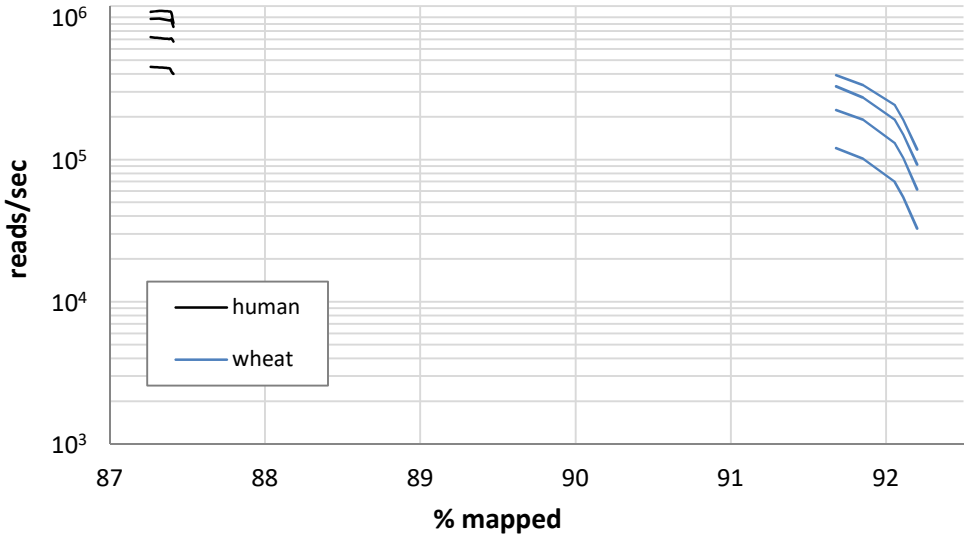
nJ	human	wheat
raw	5,875,619,304	28,516,821,874
final nJ	5,261,735,533	28,516,821,874
% pruned	10%	0%

S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

- A very brief description of short-read alignment
- Arioc: a GPU-accelerated short-read aligner
- What is a “large” genome?
- A software view of a reference genome
- Repetitiveness versus speed
- Performance

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Large genome → more work



human	ERR1347712	Simons Foundation Genome Diversity Project: SA_Kusunda_K-15_M
wheat	SRR6001710	Sequencing of flow sorted chromosome 7D from Canthatch K

- ~5x slower
  - Wheat vs human
  - Speed vs sensitivity
  - One through four V100 GPUs

Large genome → more work

- More data transferred to GPU
  - T_R/T_Q = (time to load reference sequences) ÷ (time to load reads)

	human	wheat
T_R/T_Q	1.22	13.32

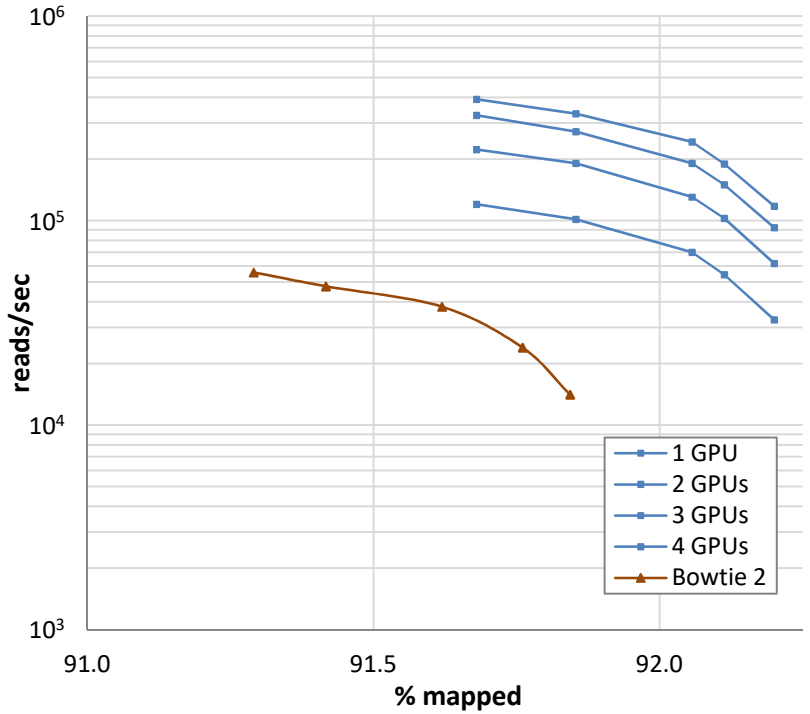
- More alignment computations per read
  - tuAlignN* = nongapped aligner (spaced seeds)
  - tuAlignG* = gapped aligner (Smith-Waterman)

	human	wheat
tuAlignN20	3.56	17.66
tuAlignN52	10.31	113.97
tuAlignGs22	1.86	31.79
tuAlignGs12	0.65	23.67
tuAlignGwn12	0.36	2.60
tuAlignGs42	0.03	0.25

S9350: CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Speed versus sensitivity

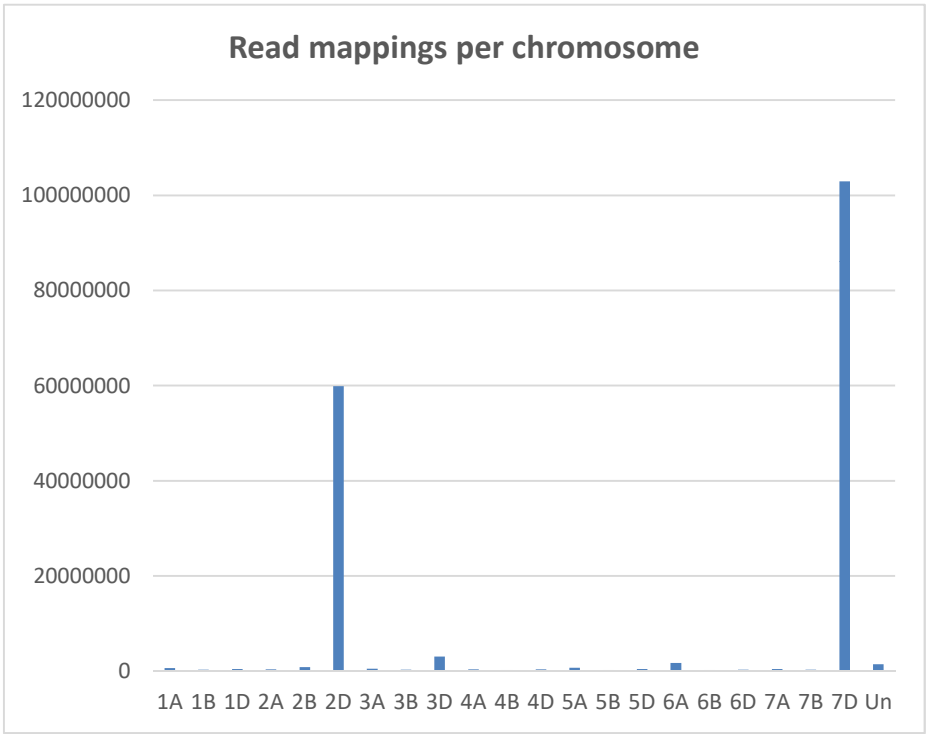
- GPU-accelerated implementation is about 10x faster than CPU-only implementation
  - Arioc (2019)
  - Bowtie 2 (2019)
- GPU-accelerated implementation scales appropriately with multiple GPUs



- Workstation configuration
  - 4 × Intel Xeon Gold 6130 CPU @ 2.10GHz (64 total threads)
  - 384GB RAM
  - CentOS 7.5.1804
  - 4 × Nvidia V100, 32GB RAM each

Accuracy

- WGS sample SRR6001710 contains DNA from a subset of the wheat genome (flow-sorted chromosome 7D)
- Distribution of mapped reads is reasonable
  - Chromosome 2D is a major contaminant (its size is very close to that of chromosome 7D)
  - Very similar to Bowtie 2



Takeaways

- A short-read aligner runs slower with a large reference genome, but not prohibitively so
- The size of the genome demands capable hardware:
  - Up to 500GB of system RAM (for building lookup tables)
  - Fast GPUs (Volta microarchitecture at a minimum)
- The repetitive nature of the genome requires careful software configuration:
  - Do not “over-optimize” the lookup tables
  - Place some of the lookup tables in GPU RAM
  - Choose optimal criteria for concordant (proper) mappings and optimal speed-vs-sensitivity tradeoff

S9350
CUDA-Accelerated Short-Read Alignment to a Large Reference Genome

Questions / Discussion

Arioc is available on Github: `https://github.com/RWilton/Arioc`

Contact: `richard.wilton@jhu.edu`