High-performance Metagenomic Data Clustering and Assembly

Kamesh Madduri

madduri@cse.psu.edu

Computer Science and Engineering

Pennsylvania State University

SIAM Annual Meeting

July 10, 2012

Talk Outline and Contributions

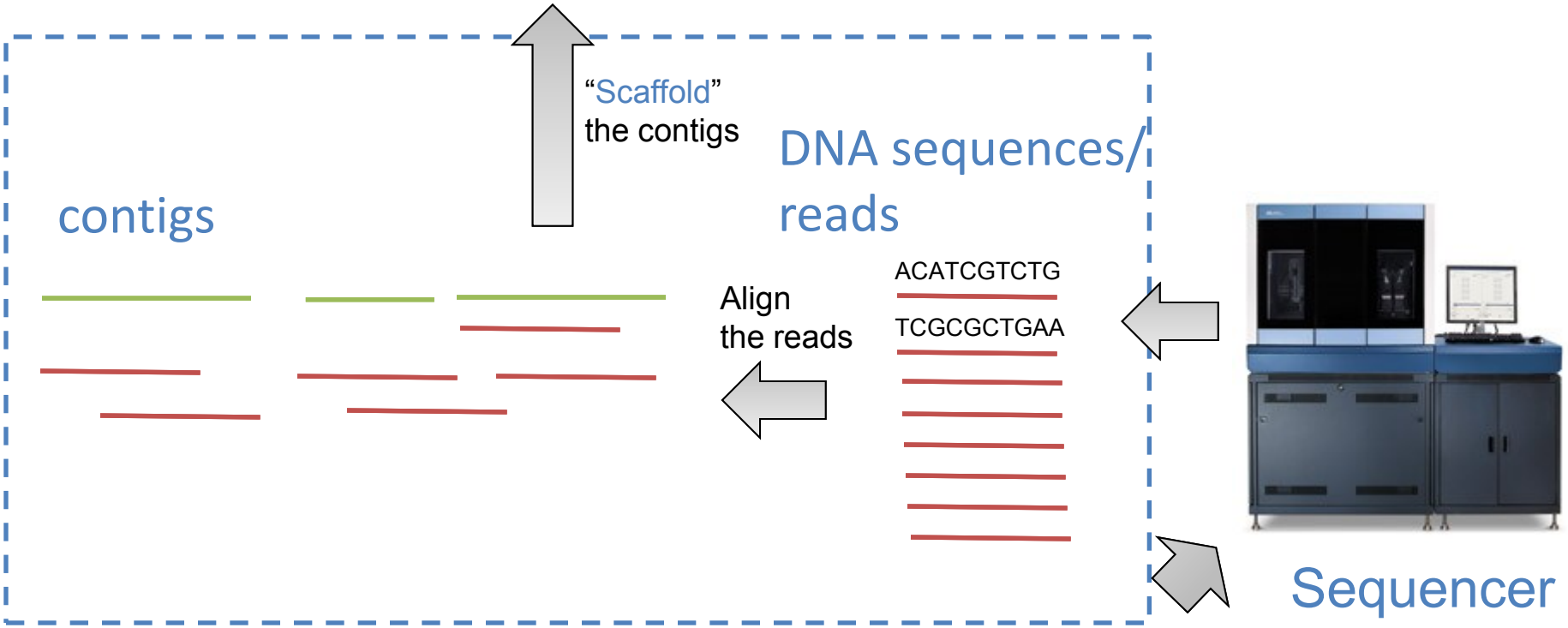
- Motivating 'big data' application: Identification of biomass-degrading genes and genomes from cow rumen

- Analysis of 268 Gbp metagenomic data
- Efficient parallelization of memory-intensive phases of de Bruijn graph-based genome assembly
- Parallel performance results
  - 150X speedup over serial approach (256-node Cray XT4 system)

Preliminaries

nucleotide

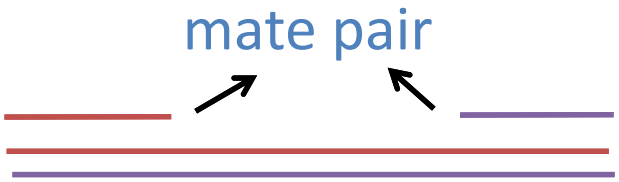
ACACGTGTGCACTACTGCACTCTACTCCACTGACTA → Genome



Sample

Next-Generation Sequencers produce "short-read" data

- New Sequencing technology (2005 -)
  - "High-throughput"
  - Illumina HiSeq 2000: 2 billion paired-end reads/run, 100 bp read length.
  - Applied Biosystems SoLiD 4: 2.25 billion reads/run, 125 bp average read length.
  - 454 GS FLX Titanium: 1 million high quality reads/run, average length of 400 bp.

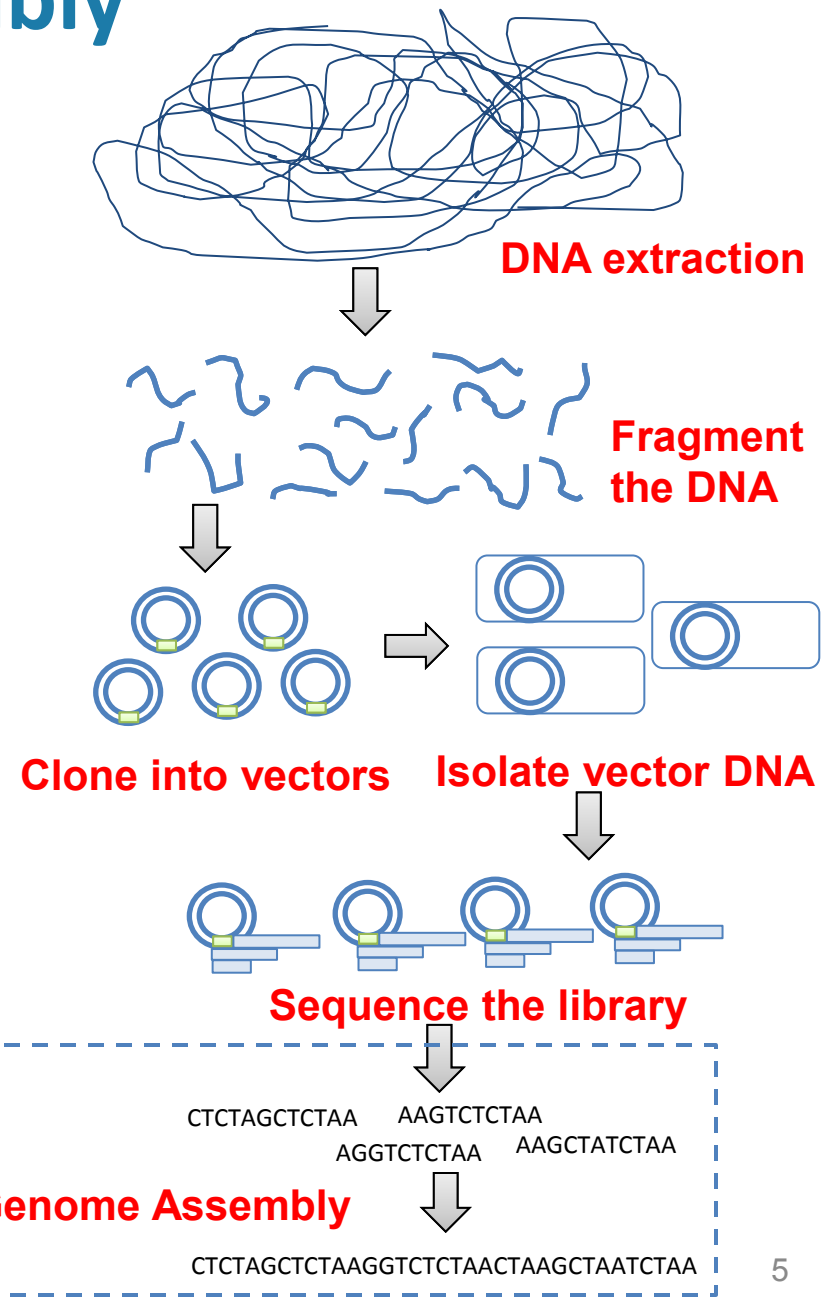


(Double-stranded) DNA fragment of known length

Image sources: www.illumina.com, www.454.com

De novo Genome Assembly

- Genome Assembly: “a big jigsaw puzzle”
- *De novo:* Latin expression meaning “from the beginning”
  - No prior reference organism
  - Computationally falls within the **NP-hard** class of problems

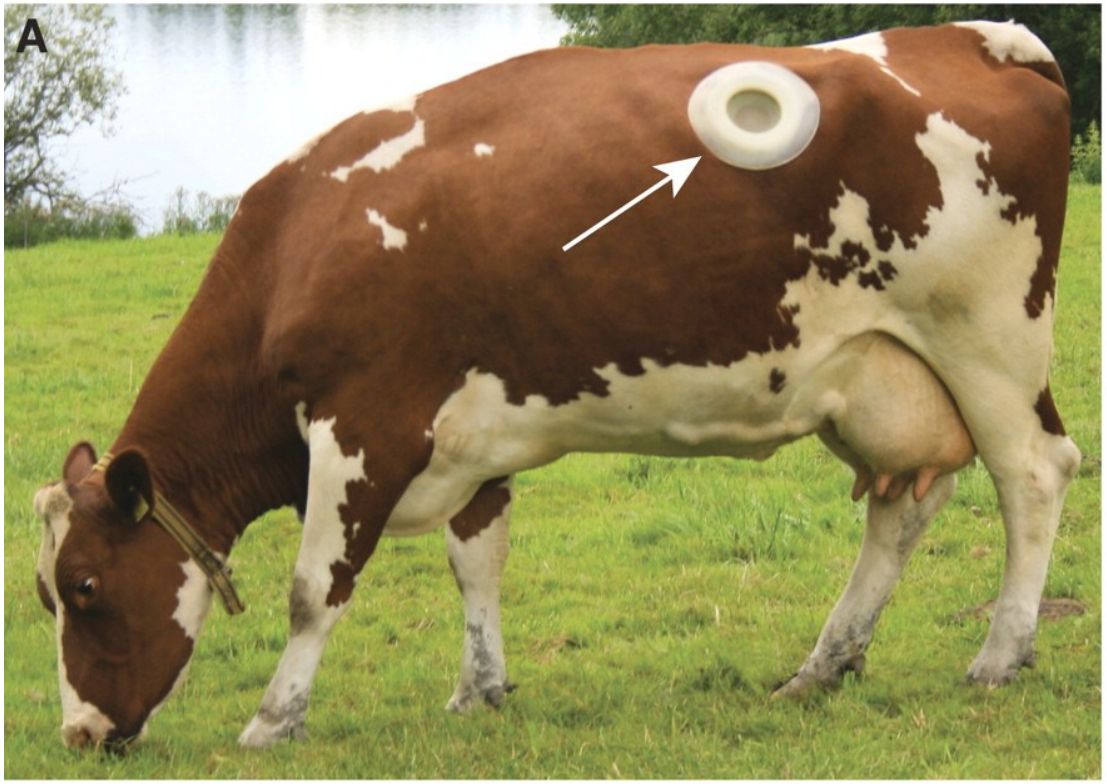


De novo Genome Assembly: Computational Challenges

- A large number of reads: millions -> billions
- Short reads
  - 1000-2000 bp with prior-generation sequencing instruments, 25-125 bp now.
- Repeats in genome
  - Complicates contig ordering
- Experimental: Sequencing errors,

  absent mate-pairs

Application: Identification of biomass-degrading Genes and Genomes from cow rumen



Goal: Identify microbial enzymes that aid in deconstruction of plant polysaccharides.
Cow rumen microbes known to be particularly effective In breaking down switchgrass.

Image Source: Hess et al., Science 331(6016), 463-467, 2011.

Metagenomes

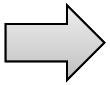
- Two major complications for de novo assembly
  - Uneven representation of organisms within a sample
  - Polymorphisms between closely related members in an environment

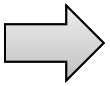
- Assembly is difficult even if we have an estimate of organism representation in a sample
- If coverage is not known, Poisson likelihood estimates used by isolate genome assemblers break down.

Eulerian path-based genome assembly strategies

- Break up the (short) reads into overlapping strings of length *k*.

k = 5

ACGTTATATATTCTA  ACGTT CGTTA GTTAT TTATA ….. TTCTA

CCATGATATATTCTA  CCATG CATGA ATGAT TGATA ….. TTCTA

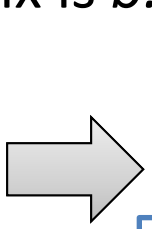
- Construct a de Bruijn graph (a directed graph representing overlap between strings)

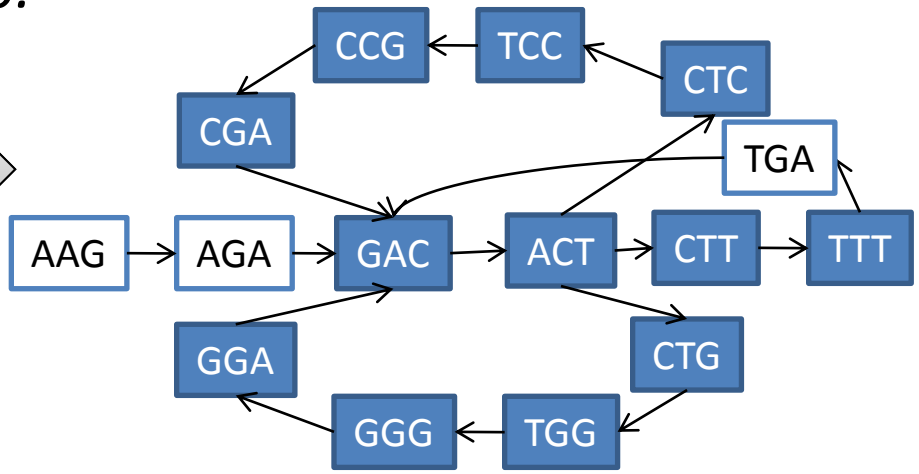
de Bruijn graphs

- Each (*k-1)*-mer represents a node in the graph
- Edge exists between node *a* to *b* iff there exists a *k*-mer such that its prefix is *a* and suffix is *b.*

AAGACTCCGACTGGGACTTT

ACTCCGACTGGGACTTTGAC





- Traverse the graph (if possible, identifying an *Eulerian path*) to form contigs.
- However, correct assembly is just one of the many possible Eulerian paths.

Algorithms and Software based on this approach

- Velvet (EBI)
- Meraculous (JGI)
- ABySS (Canada Genome Sciences Center)
- ALLPATHS (Broad Institute)
- YAGA (Iowa State)
- SOAPdenovo (Beijing Genome Institute)
- Contrail (Univ of Maryland)
- Euler-SR (UCSD)
- ….

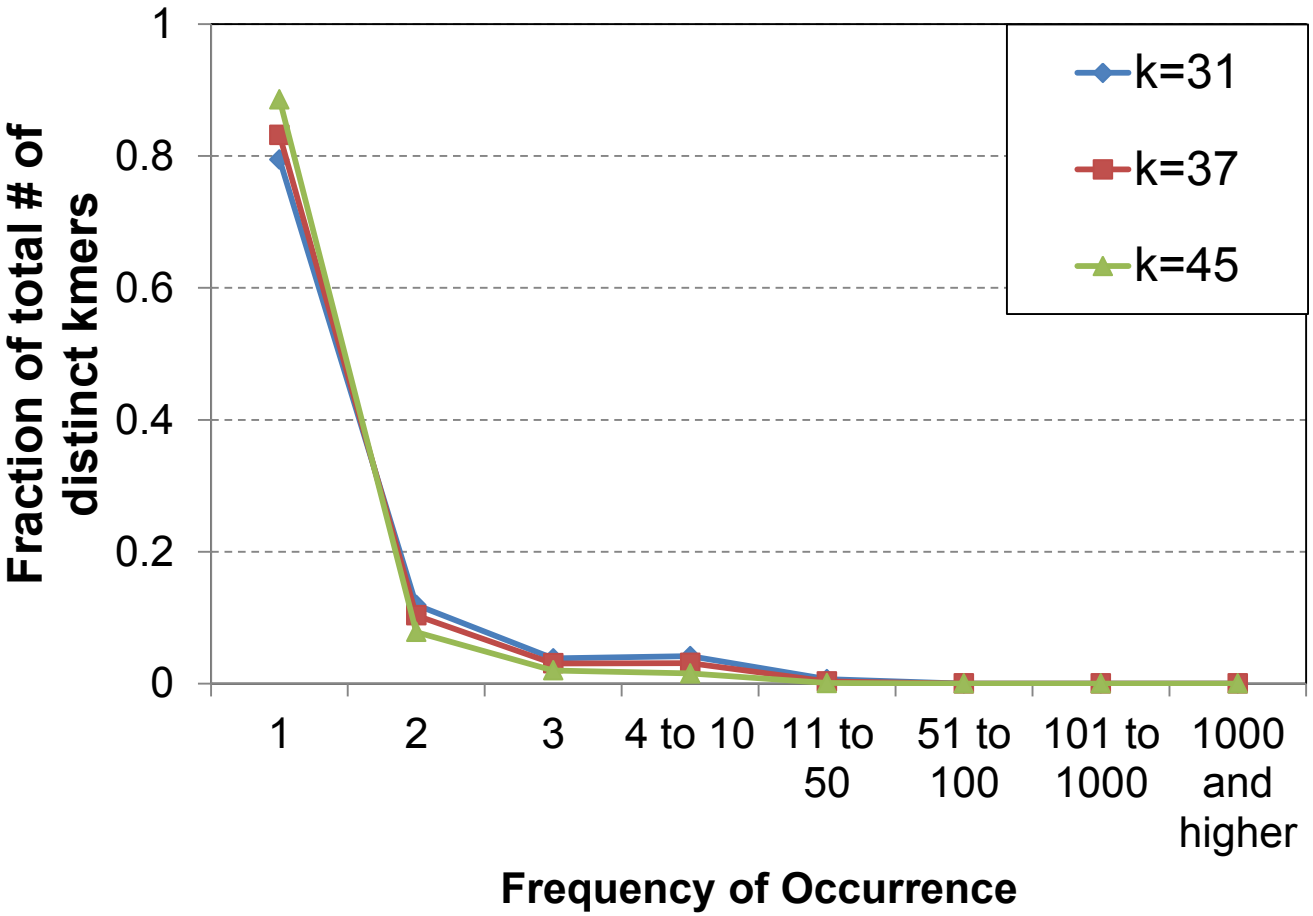
JGI “cow rumen” dataset: Challenges

- Metagenome composition and coverage estimate not known
- Data size quite large
  - 268 Gb, 1.2 billion paired-end reads
  - Velvet (serial execution) requires 2+ TB memory
- Unclear on quality assessment of resulting assembly
- Difficulty experimenting with existing software
  - what parameters to use
  - what routines to change for metagenomes?
  - Sub-optimal data representation

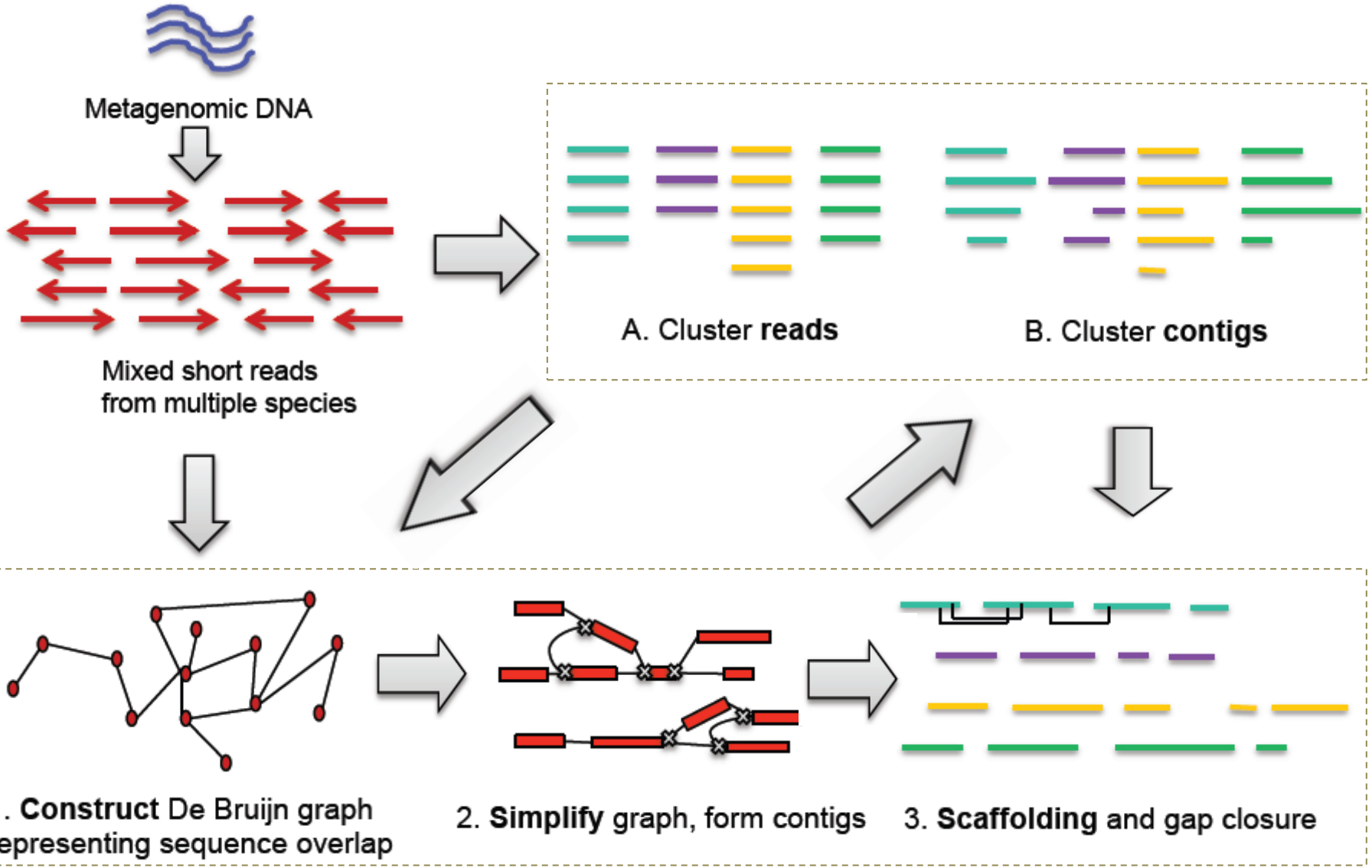
Kmer spectrum

- What values of *k* are appropriate for this data set?
- If the data is error-free, # of unique *k*mers should be bounded by length of genome.
- Experimented with different values of *k*
  - 31, 37, 45 (note: read length is 125)
- Surprising results
  - 80% of total enumerated kmers are unique
  - High percentage of kmers that occur just once
  - Very low coverage?

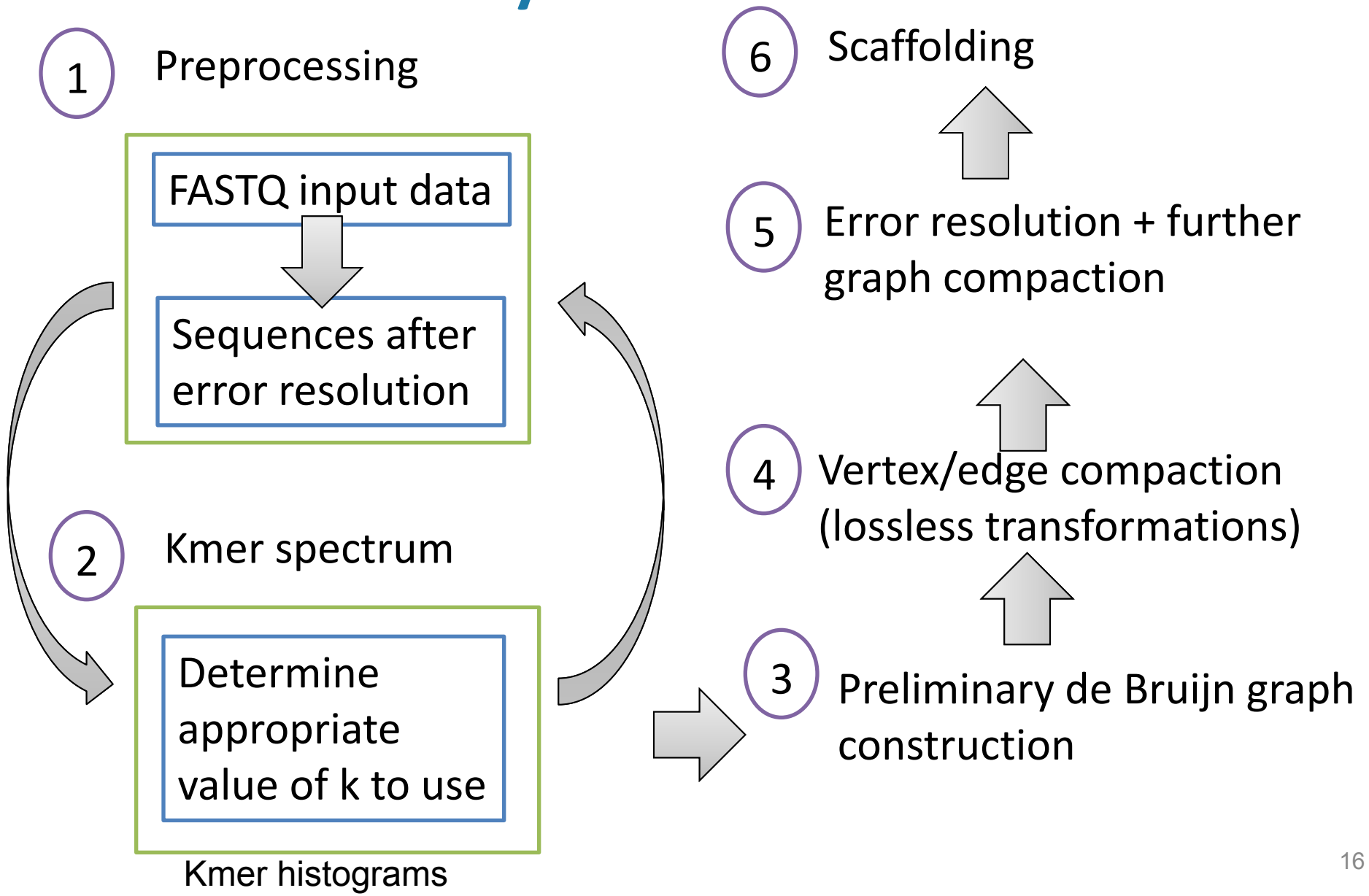
Kmer frequency spectrum



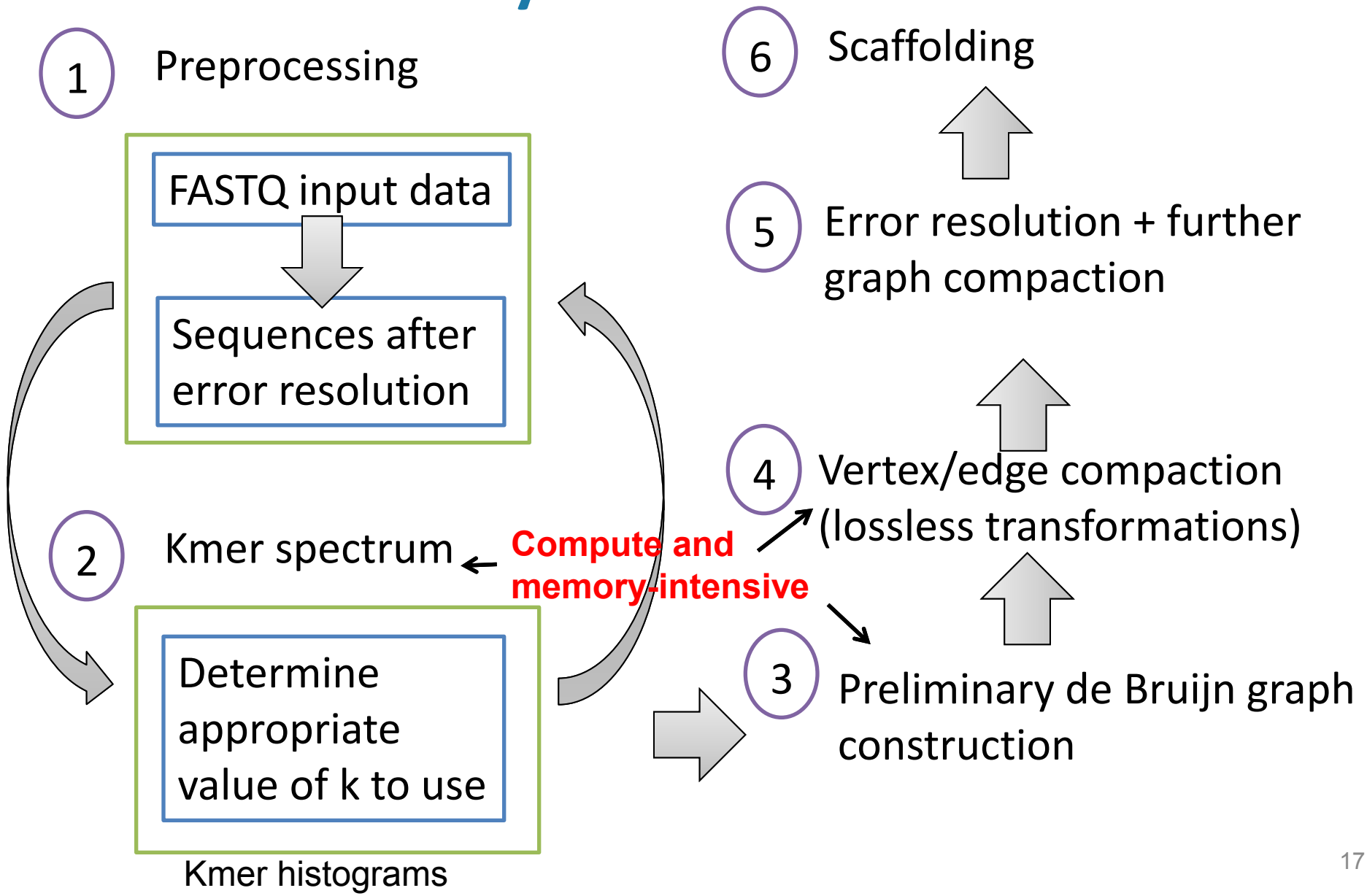
Assembly and Clustering Methodology



Breakdown of parallel de Bruijn graph-based assembly scheme

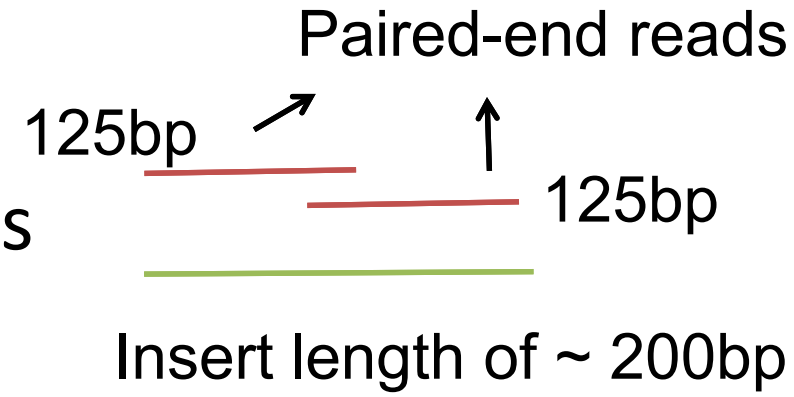


Breakdown of parallel de Bruijn graph-based assembly scheme



1. Preprocessing

- Process base quality information
- Mark ambiguous bases
- Try to merge paired reads



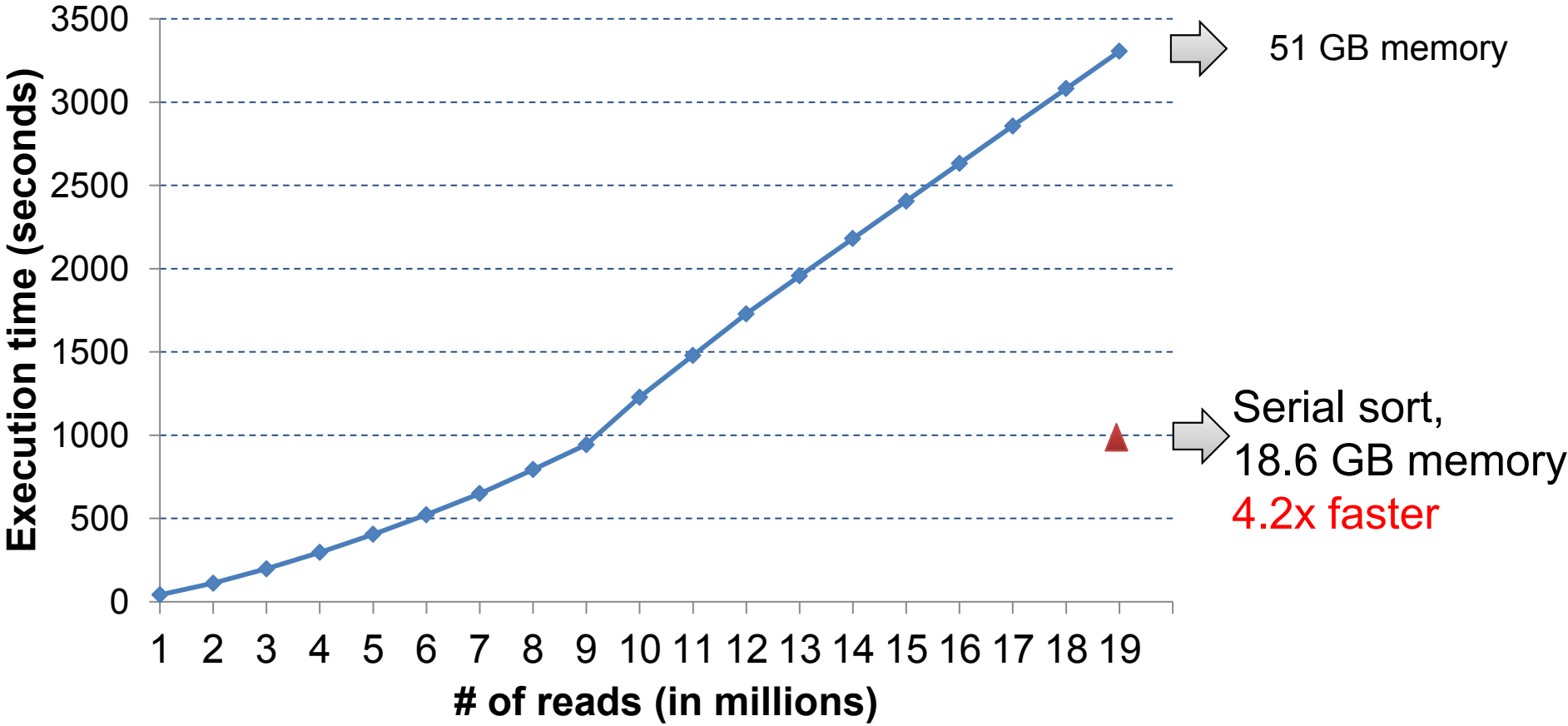
- Write back filtered reads
- Parallelization strategy: split input files into "P" parts; each node processes its file independently
  - Predominantly I/O bound

2. Kmer spectrum construction

- Need a dictionary to track occurrences of each kmer
- Hashing expensive for large data sizes; maintaining an ordered set unnecessary when updates are predominantly insert-only (“cow rumen” dataset, large “k”)
- Alternative: Ingest all kmers, perform lexicographical sort
- Parallelization: enumerate kmers independently + one global sort to get kmer count

Finding unique kmers: hashing vs sorting

Splay tree update time for a data set of 19.5 million (125 bp) reads (k=61)

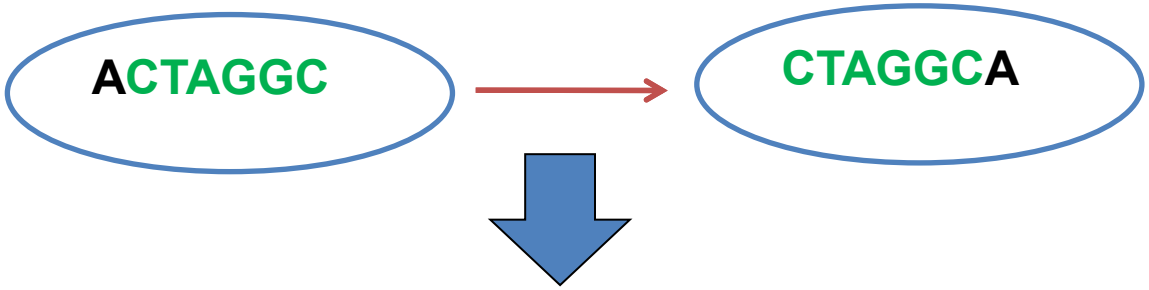


Serial performance results on a 512 GB system (2.6 GHz Opteron processor)

3. Graph construction

- Store edges only, represent vertices (kmers) implicitly.
- Distributed graph representation
- Sort by start vertex
- Edge storage format:

Read ‘x’:



Store edge (ACTAGGCA), orientation, originating read id (x), edge count

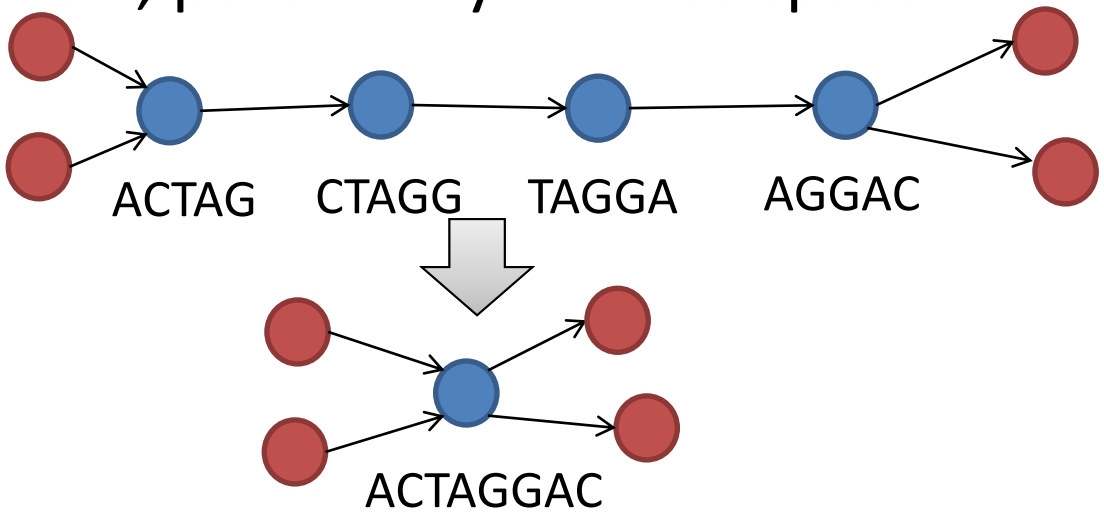
Use 2 bits per nucleotide

4. Vertex compaction

- High percentage of unique kmers

  $\Rightarrow$ Try compacting kmers from same read first

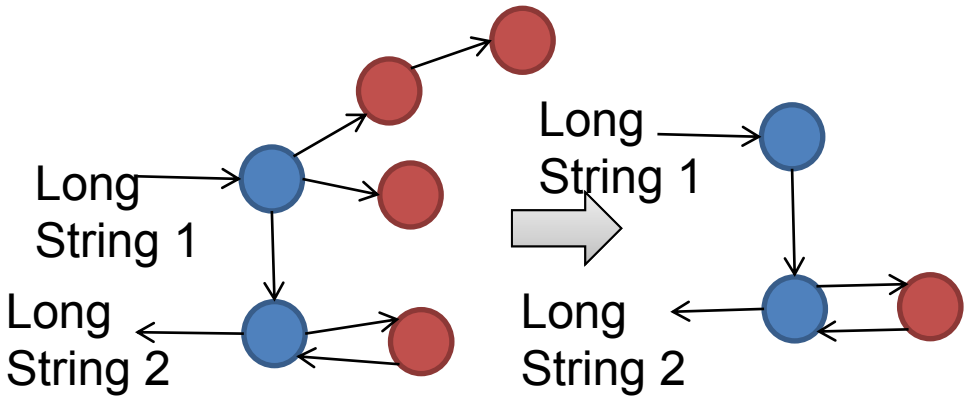
  – If kmer length is k, potentially k-times space reduction!



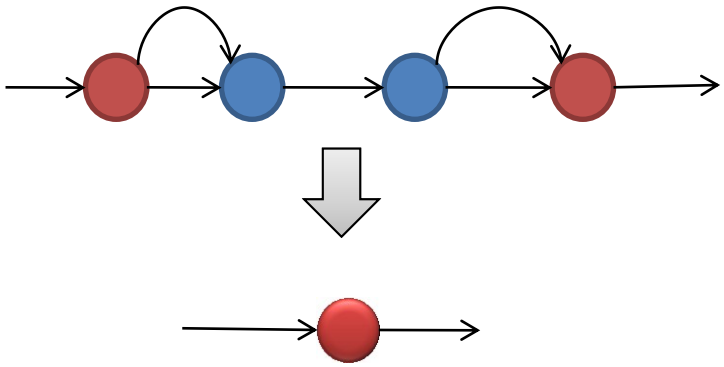
- Parallelization: computation can be done locally after sorting by read ID

Metagenome-specific steps

- Split ‘high-degree’ vertices
- Identify connected components
- Error resolution and scaffolding can be concurrently performed on multiple independent components



Compress/remove whiskers

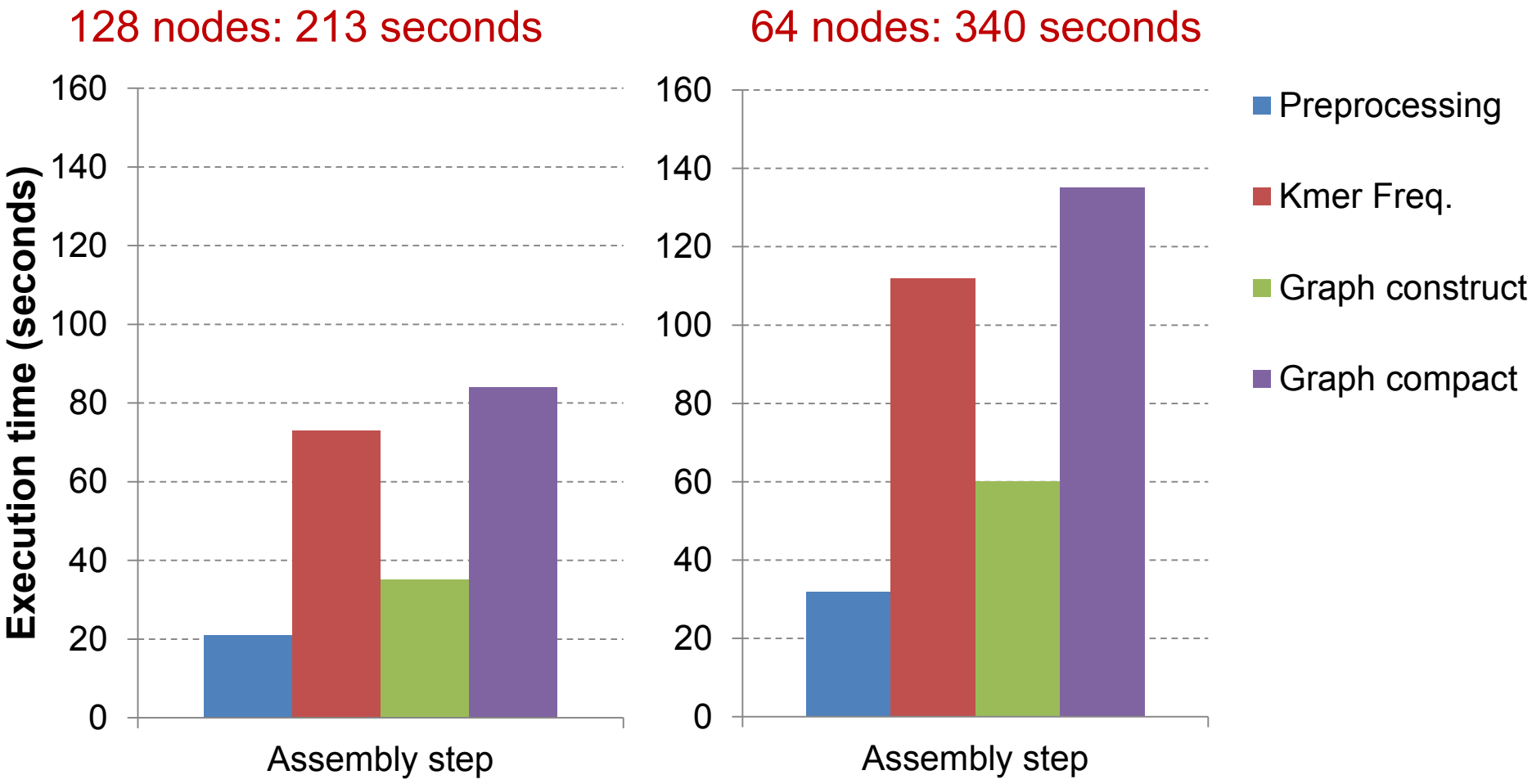


Identify and fix “low coverage” edges

Parallel Implementation Details

- Current data set (after preprocessing) requires 320 GB for in-memory graph construction
  - Experimented with 64 nodes (256-way parallelism) and 128 nodes (512-way) of NERSC Franklin (Cray XT4 system, 2.3 GHz quad-core Opteron processor)
- MPI across nodes + OpenMP within a node
- Local sort: multicore-parallel quicksort
- Global sort: sample sort

Parallel Performance



- Comparison: *Velveth* (up to graph construction) takes ~ 12 hours on the 512 GB Opteron system.

Talk Summary

- Overview of the de novo genome assembly problem for short-read sequence data
- Outlined components of a de Bruijn graph-based assembler customized for metagenomic data
- Significant performance improvement over serial state-of-the-art approach
  - **150x faster** at **256-way** node concurrency on a Cray XT4 system

Acknowledgments

- Shruthi Prabhakara, Raj Acharya, Penn State
- M. Poss, M. Roossinck, Penn State
- Alex Sczyrba, Rob Egan, Jarrod Chapman, Kostas Mavromatis, DOE Joint Genome Institute
- Victor Markowitz, John Shalf, Kathy Yelick, Lawrence Berkeley National Laboratory

Thank you!

Questions?