Parallel and Memory-efficient Preprocessing for Metagenome Assembly

Vasudevan Rengasamy Paul Medvedev Kamesh Madduri

School of EECS
The Pennsylvania State University

{vxr162, pashadag, madduri}@cse.psu.edu

HiCOMB 2017

Talk Outline

- Motivation for our work
- MetaPrep: a new metagenome pre-processing strategy
- MetaPrep evaluation
  - Parallel Scaling
  - Comparison to prior work
  - Impact on Metagenome assembly
- Conclusions and Future work

Metagenome assembly

What is metagenome assembly?

- Metagenome: Mixed genomes present in an environment sample (Soil, Human gut, etc.).
- Assembly: Re-constructing genome sequence from **reads**.

Metagenome assembly

What is metagenome assembly?

- Metagenome: Mixed genomes present in an environment sample (Soil, Human gut, etc.).
- Assembly: Re-constructing genome sequence from **reads**.

Why is metagenome assembly challenging?

1. Uneven coverage of genomes.
2. Repeated sequences across genomes.
3. Variable sizes of genomes.
4. Large dataset sizes (as the output from multiple sequencing runs may be merged).

Metagenome assembly tools (MEGAHIT, MetaVelvet, metaSPAdes, etc.) attempt to overcome these challenges.

MEGAHIT [Li2016] metagenome assembler

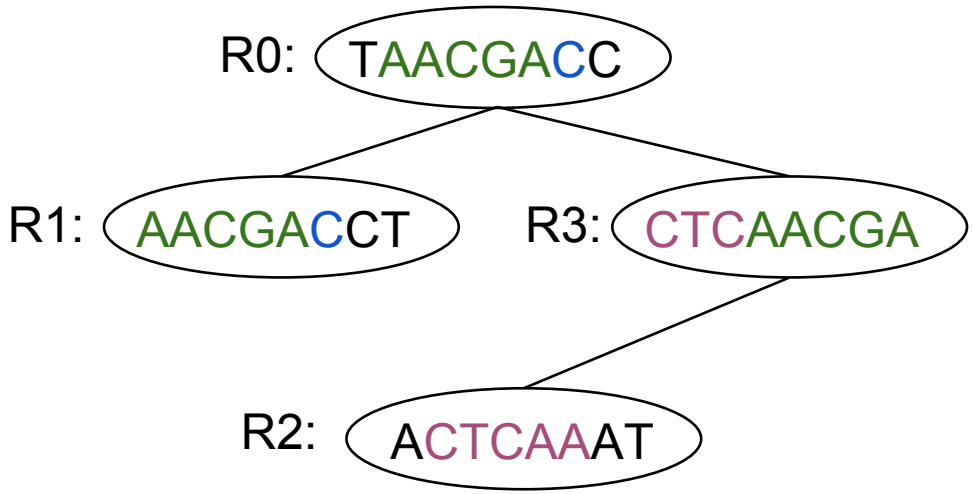
- State-of-the-art metagenome assembler.
- Uses a highly compressed de Bruijn graph representation.
- Refines assembly quality by using multiple *k*-mer lengths.
- Supports single-node shared memory parallelism (both CPUs and GPUs).
- 47 minutes to assemble a metagenome dataset containing 4.26 Gbp.

A preprocessing strategy for Metagenome assembly

- Introduced by Howe et al. [Howe2014].
- After filtering low frequency *k*-mers, partition de Bruijn graph into weakly connected components (WCCs).
- Assemble each large component independently.

Recent work on metagenome partitioning [Flick2015]

- Construct an undirected **read graph** instead of a de Bruijn graph.
- Find connected components in the read graph using a distributed memory parallel approach based on Shiloach-Vishkin algorithm.
- Read graph components correspond to de Bruijn graph WCCs.



Our contributions

- Novel multi-stage algorithm to find connected components from read graphs.
- End-to-end hybrid parallelism using MPI and OpenMP.
- Memory aware implementation.
- Evaluate impact of preprocessing on metagenome assembly.

MetaPrep

- New **Meta**genome **Prep**rocessing tool.
- Main memory use is parameterized.
  - Multipass approach: Only enumerate a subset of k-mers in each pass.
  - e.g., 10 passes $\Rightarrow$ 10$\times$ memory reduction.
- $log(P)$ inter-node communication steps.

Talk Outline

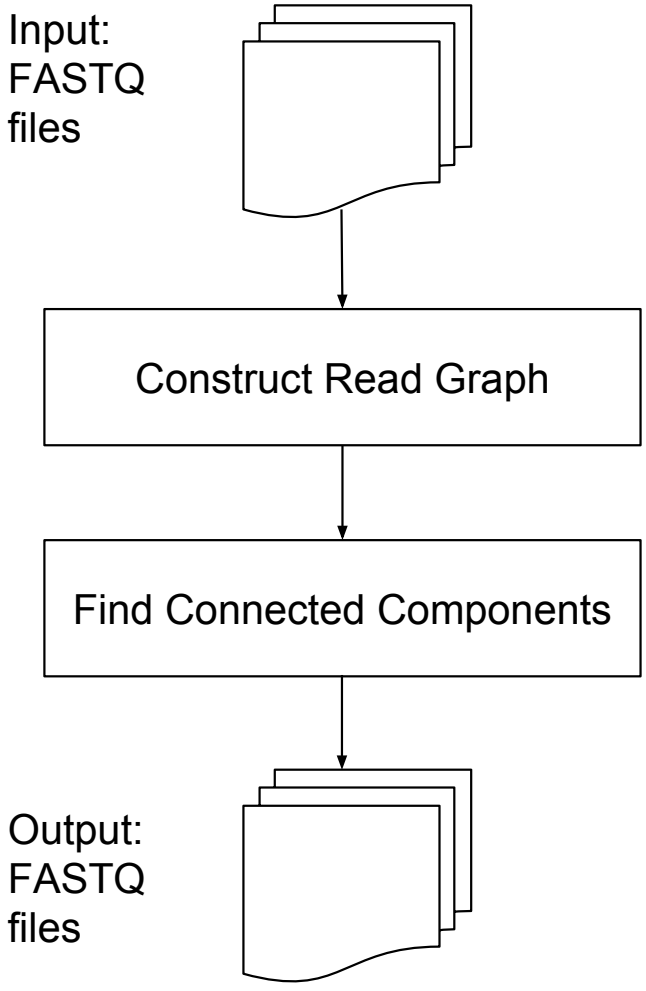
Motivation for our work

MetaPrep: a new metagenome pre-processing strategy

MetaPrep evaluation
- Parallel Scaling
- Comparison to prior work
- Impact on Metagenome assembly

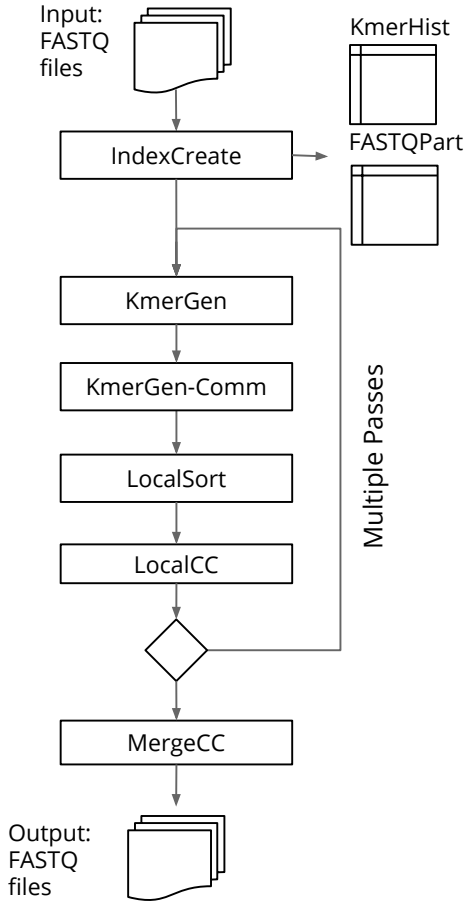
Conclusions and Future work

MetaPrep overview



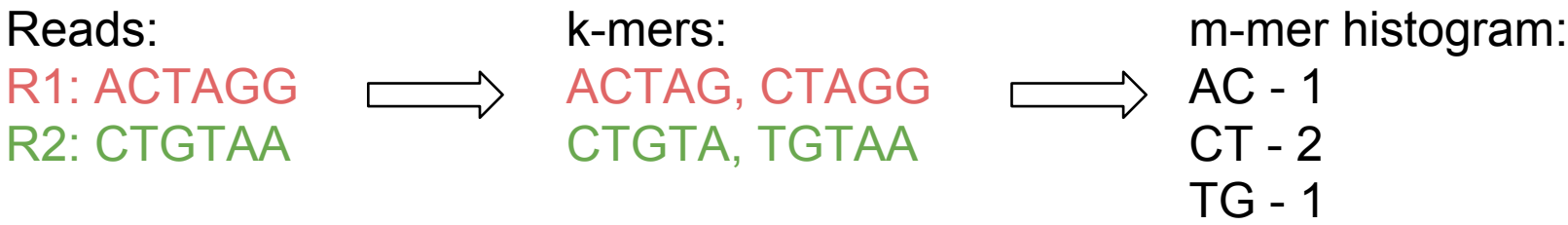
MetaPrep overview

	MetaPrep step	Function
	IndexCreate	Create index files for parallel runs.
1	KmerGen	Enumerate $\langle k$-mer, read$_i\rangle$ tuples.
2	KmerGen-Comm	Transfer $\langle k$-mer, read$_i\rangle$ tuples to owner tasks.
3	LocalSort	Sort tuples by k-mers.
4	LocalCC	Identify connected components (CCs).
5	MergeCC	Merge components across tasks, create output FASTQ files with reads from largest CC and other CCs.



A simple strategy for static work partitioning

- Precompute an m-mer histogram ($m \ll k$, defaults are $k = 27$, $m = 10$)
- Used to partition k-mers across MPI tasks and threads in a load balanced manner.

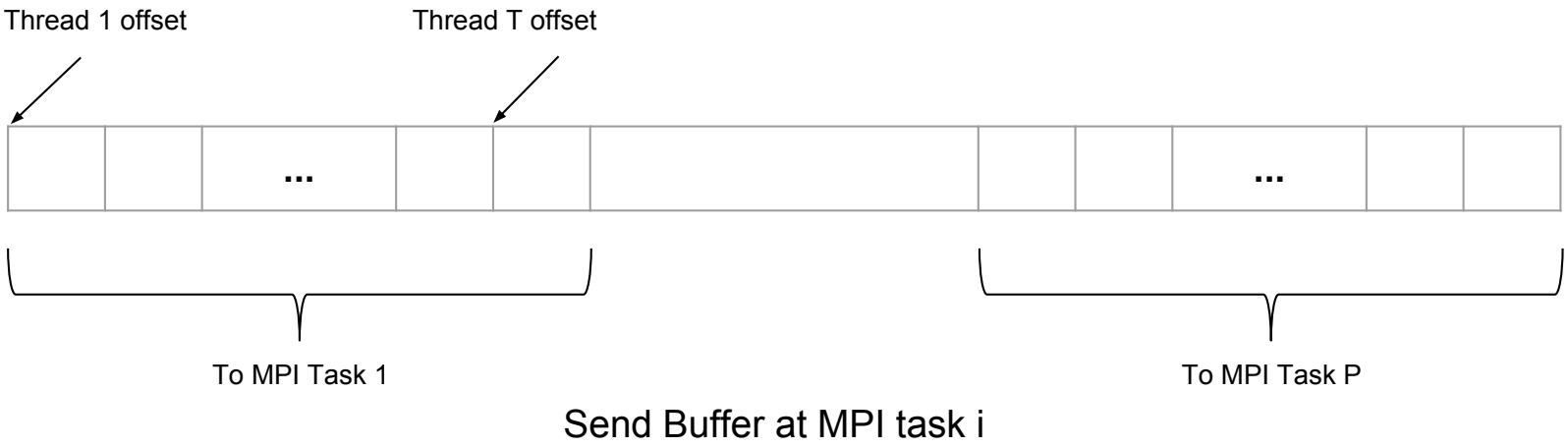


Notation

Notation	Description
M	Total number of k-mers enumerated
R	Paired-end read count
S	Number of I/O passes
P	Number of MPI tasks
T	Number of threads per task

k-mer Enumeration

- Generate $\langle k\text{-mer}, \text{read\_id} \rangle$ tuples.
- Multiple threads write to single array without synchronization. Offsets precomputed.
- Output: a buffer on each MPI task.
  - k-mers are partially sorted.
- Time: $O(\frac{MS}{PT})$, Space $\approx \frac{24M}{SP}$ bytes.

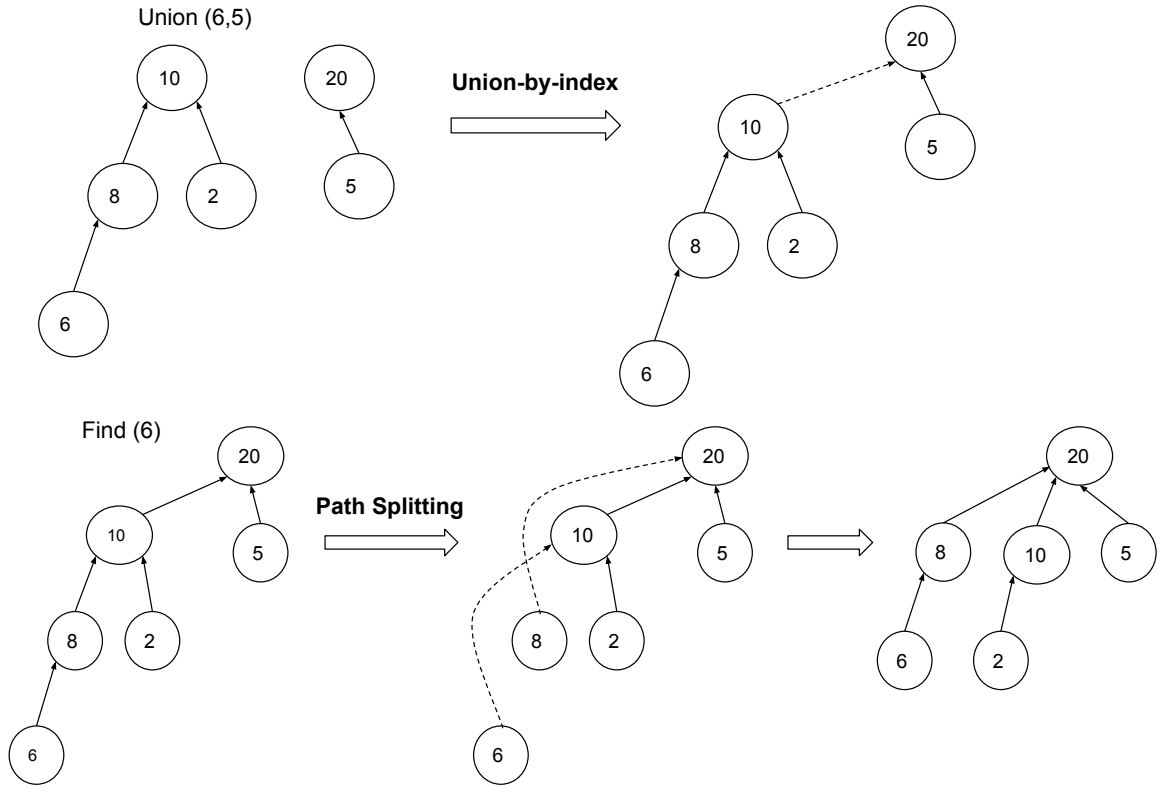


Sort by *k*-mer

- Sort tuples by *k*-mer value to identify reads with common *k*-mer and create read graph edges.
- Radix sort implementation.
  - Reuse send buffer $\Rightarrow$ No additional memory .
  - Partition tuples into T disjoint ranges.
  - Sort ranges in parallel using T threads.
- Time: $O(\frac{M}{PT})$, Space $\approx \frac{24M}{SP}$ bytes.

Identify connected components

- Find connected components using edges from local *k*-mers.
- Union-by-index and path splitting.

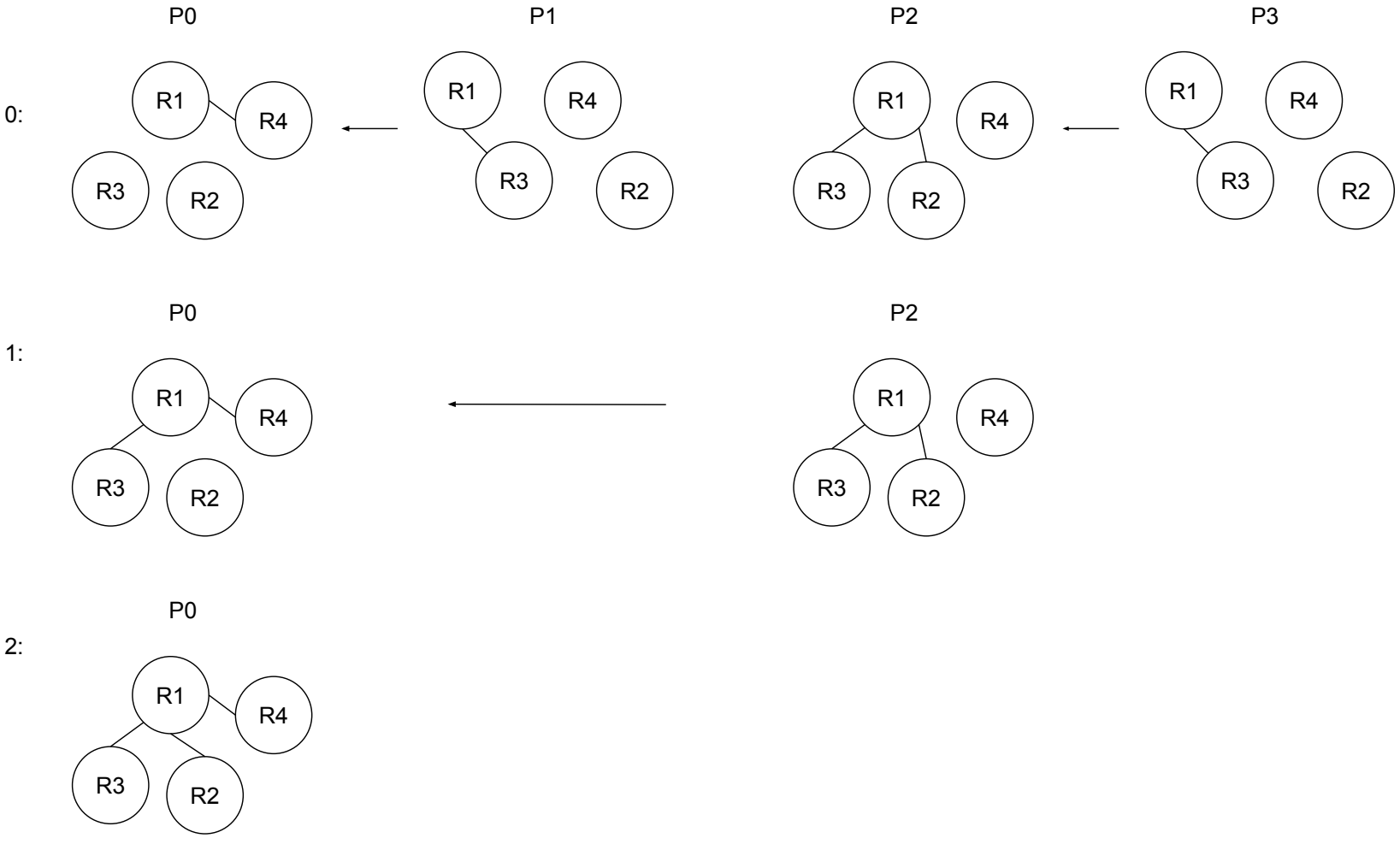


Identify connected components

- Find connected components using edges from local k-mers.
- Union-by-index and path splitting.
- No critical sections.
  - Store edges that merges components (similar to [Patwary2012]).
  - Process edges again in case of lost updates.
- Time: $O(\frac{M}{PT} log^* R)$, Space $\approx \frac{12M}{SP} + 4R$ bytes.

Merge components

- Merge component forests in each MPI task in $log\ P$ iterations.
- Time: $O(R\ log\ P\ log^* R)$, Space $\approx 8R$ bytes.



Talk Outline

Motivation for our work

MetaPrep: a new metagenome pre-processing strategy

MetaPrep evaluation
- Parallel Scaling
- Comparison to prior work
- Impact on Metagenome assembly

Conclusions and Future work

Experiments and Results

Description of datasets

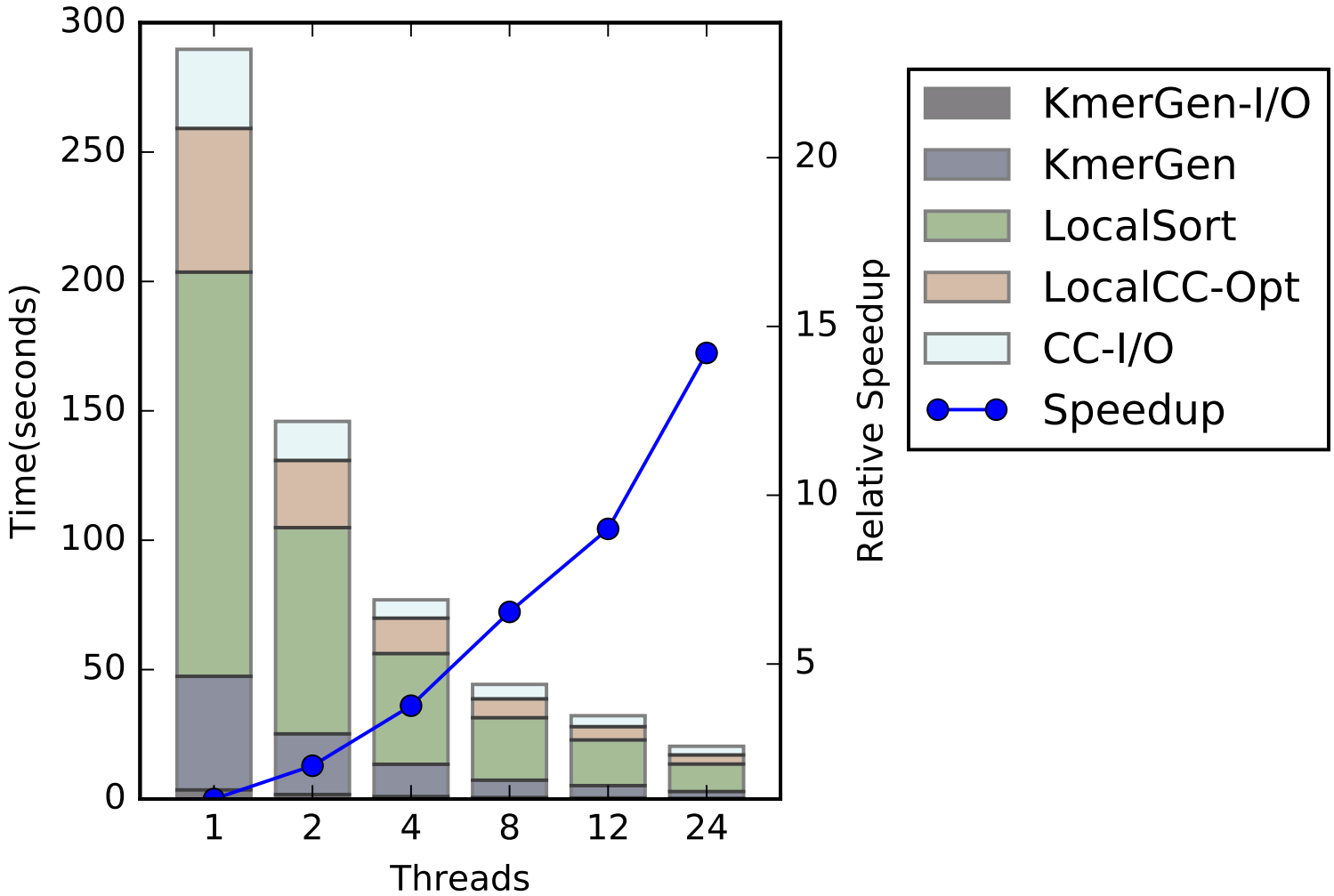
ID	Dataset	Read Count R ($\times 10^6$)	Size (Gbp)	Source
HG	Human gut	12.7	2.29	NCBI (SRR341725)
LL	Lake Lanier	21.3	4.26	NCBI (SRR947737)
MM	Mock microbial community	54.8	11.07	NCBI (SRX200676)
IS	Iowa, Continuous corn soil	1132.8	223.26	JGI (402461)

Machine configuration

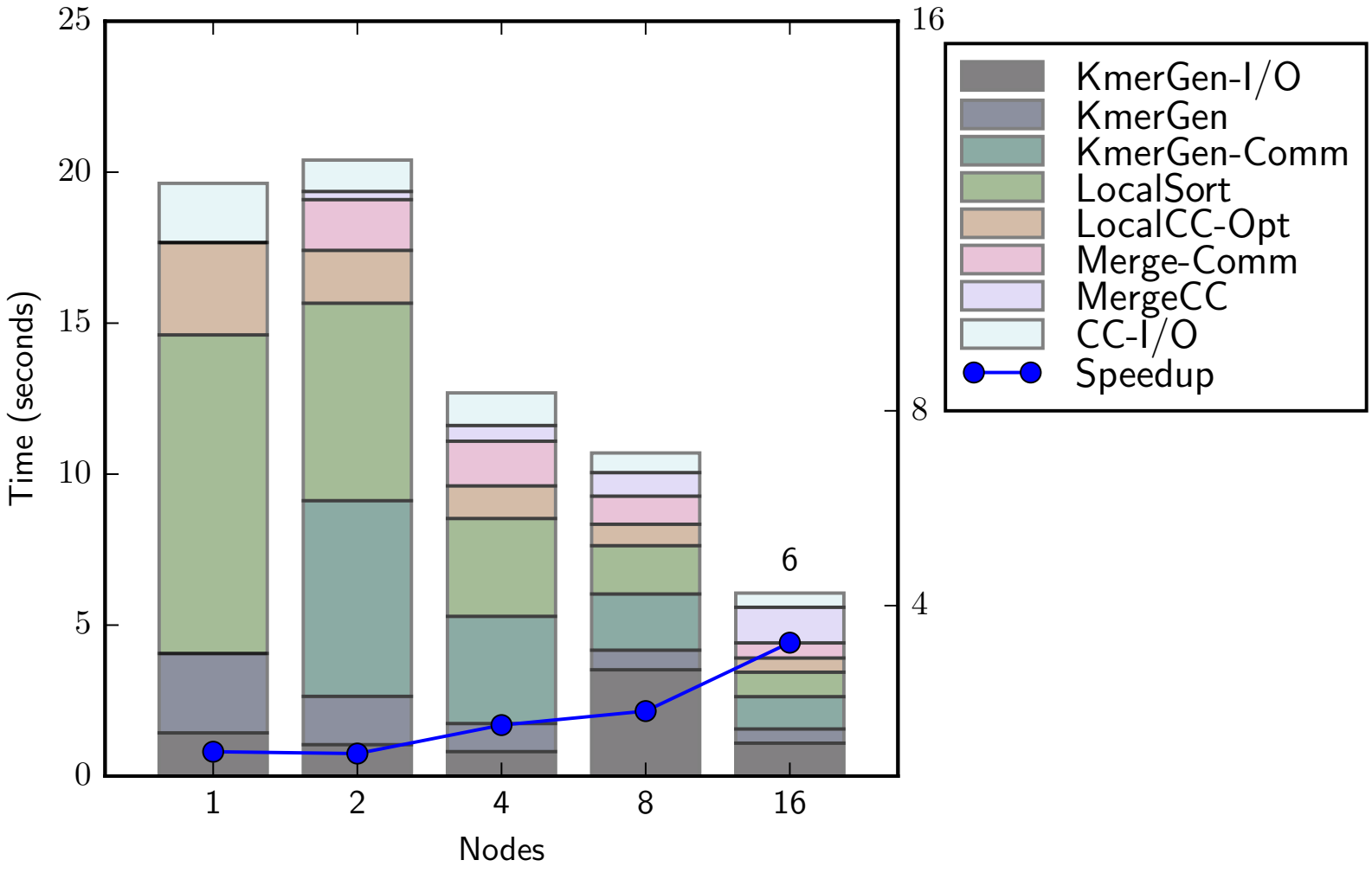
- Edison supercomputer at NERSC
  - Each node has 2$\times$ 12-core Ivy bridge processors and 64 GB memory.

Overview

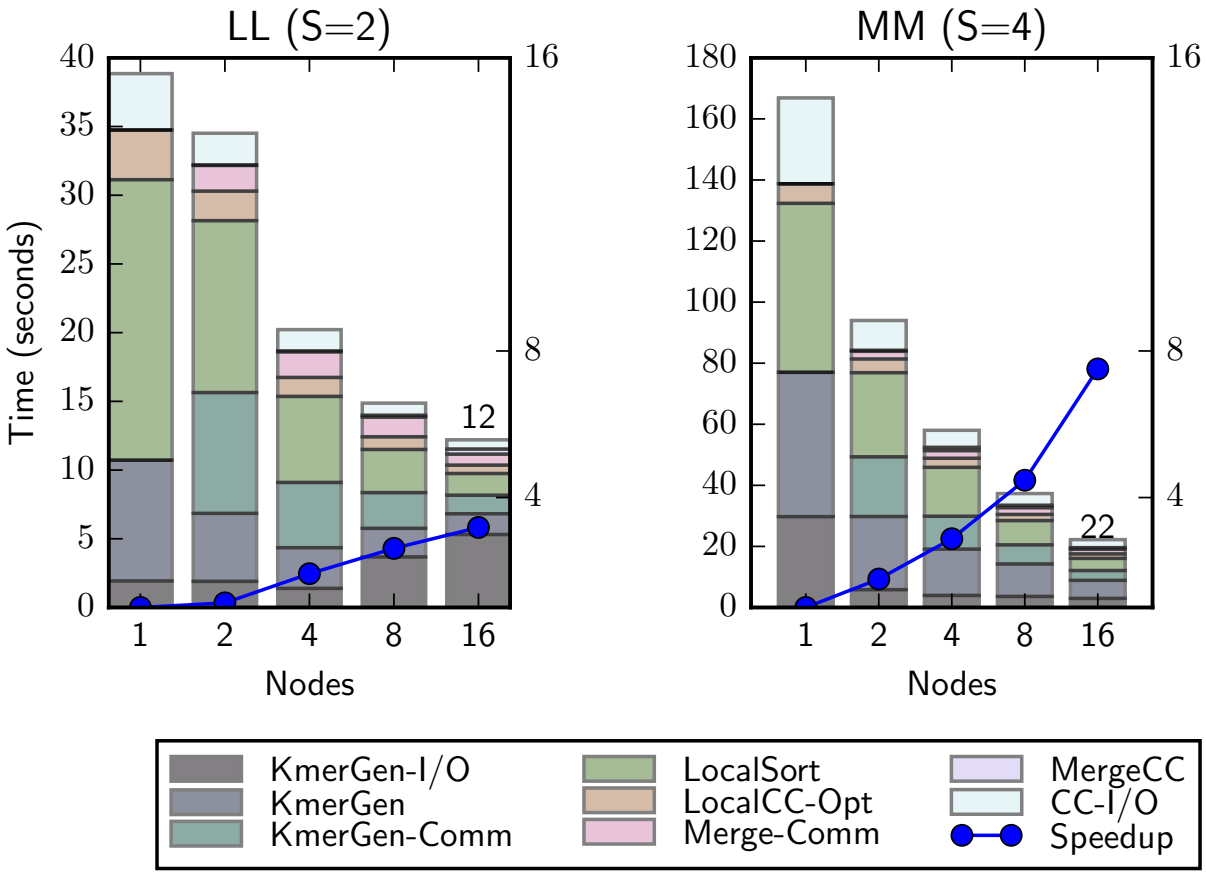
Single node scaling for Human Gut (HG) Dataset



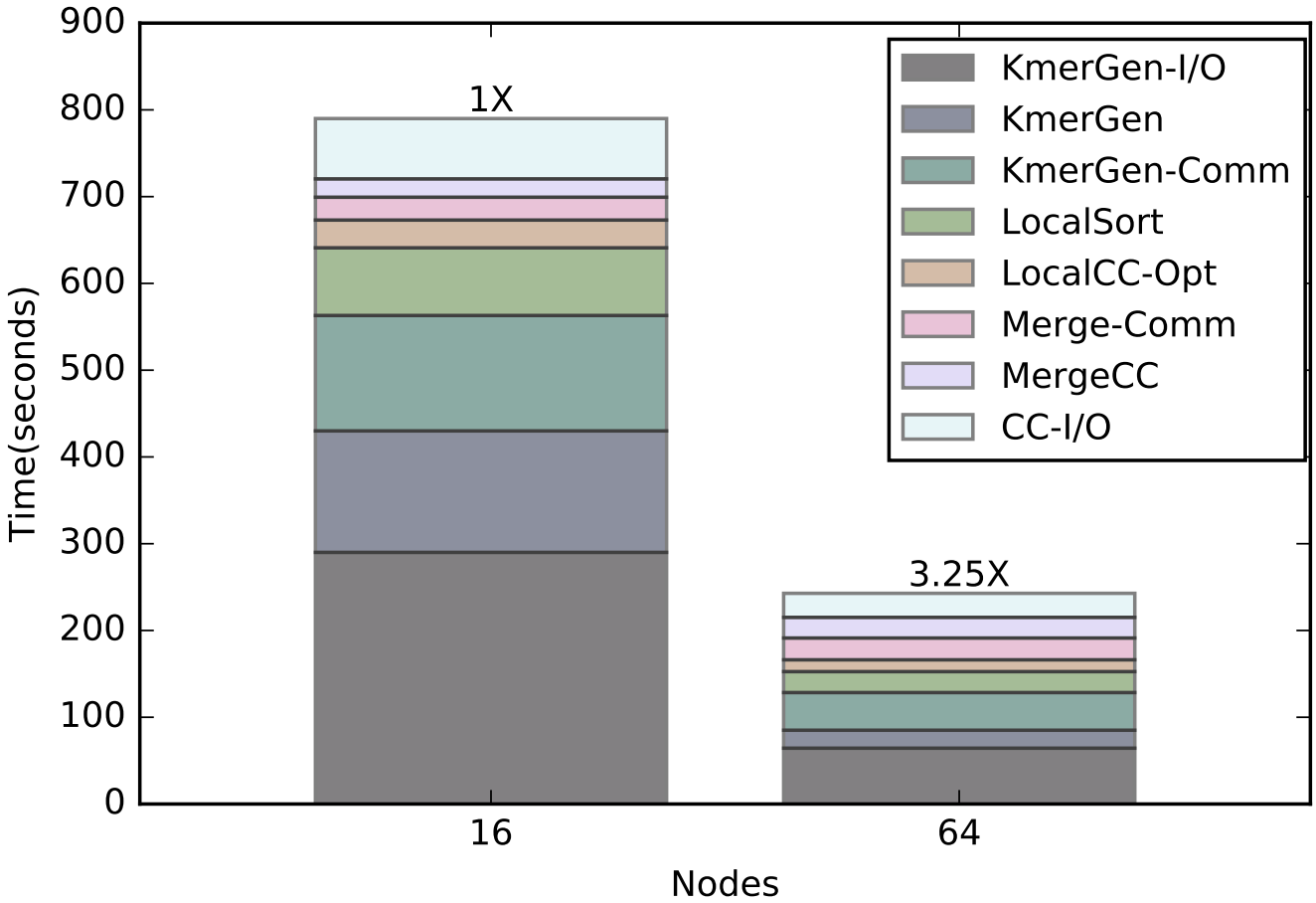
Multi-node scaling for Human Gut (HG) Dataset



Multi-node scaling for LL and MM datasets



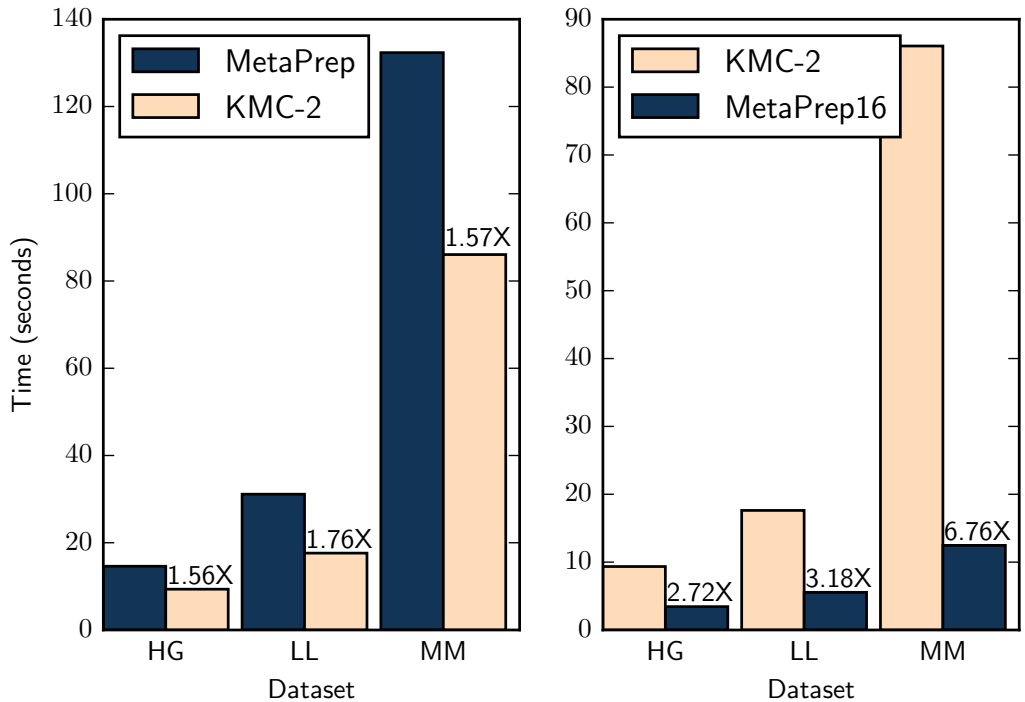
Multi-node scaling for Iowa Continuous Soil dataset



For 16 node run, S = 8. For 64 node run, S = 2.

Overview

KmerGen performance comparison with KMC-2 *k*-mer counter [Deorowicz2015]



- MetaPrep16: MetaPrep run using 16 nodes.

Comparison with read graph connectivity [Flick2015]

Table 1: Execution time comparison with Metagenome partitioning work (AP_LB) using 16 nodes.

Dataset	Time (seconds)		MetaPrep Speedup
	MetaPrep	AP_LB	
HG	5.5	23.6	4.22×
LL	11.5	25.9	2.25×
MM	19.6	56.1	2.86×

- 21 iterations for AP_LB vs 4 for MetaPrep for MM dataset.

Overview

Motivation for our work

MetaPrep: a new metagenome pre-processing strategy

MetaPrep evaluation

- Parallel Scaling
- Comparison to prior work
- Impact on Metagenome assembly

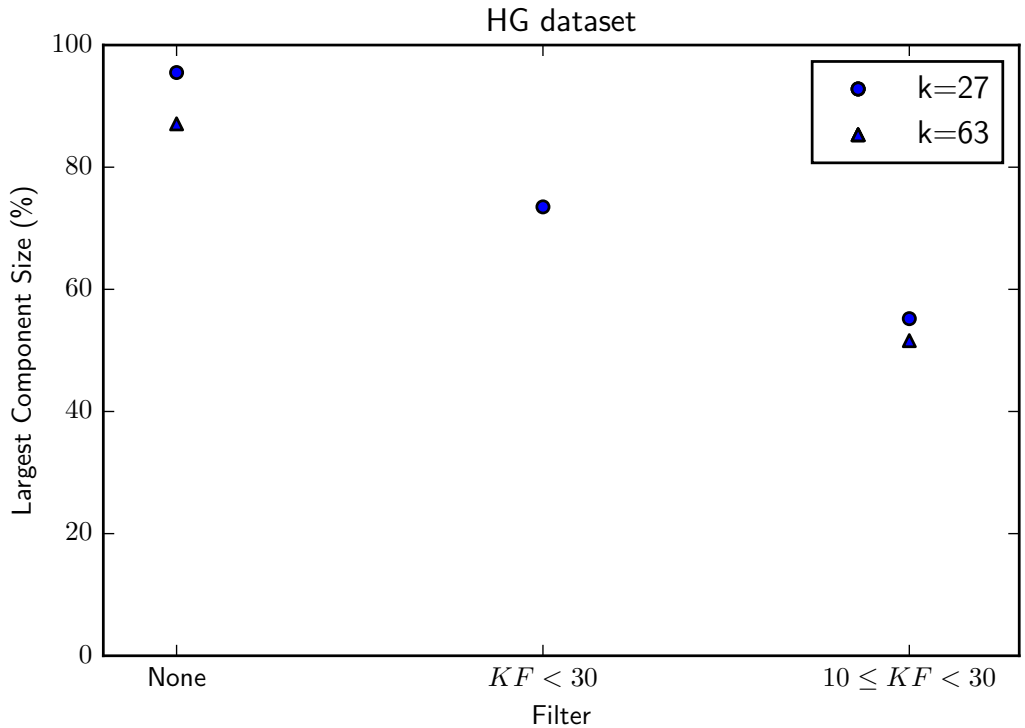
Conclusions and Future work

Largest component size

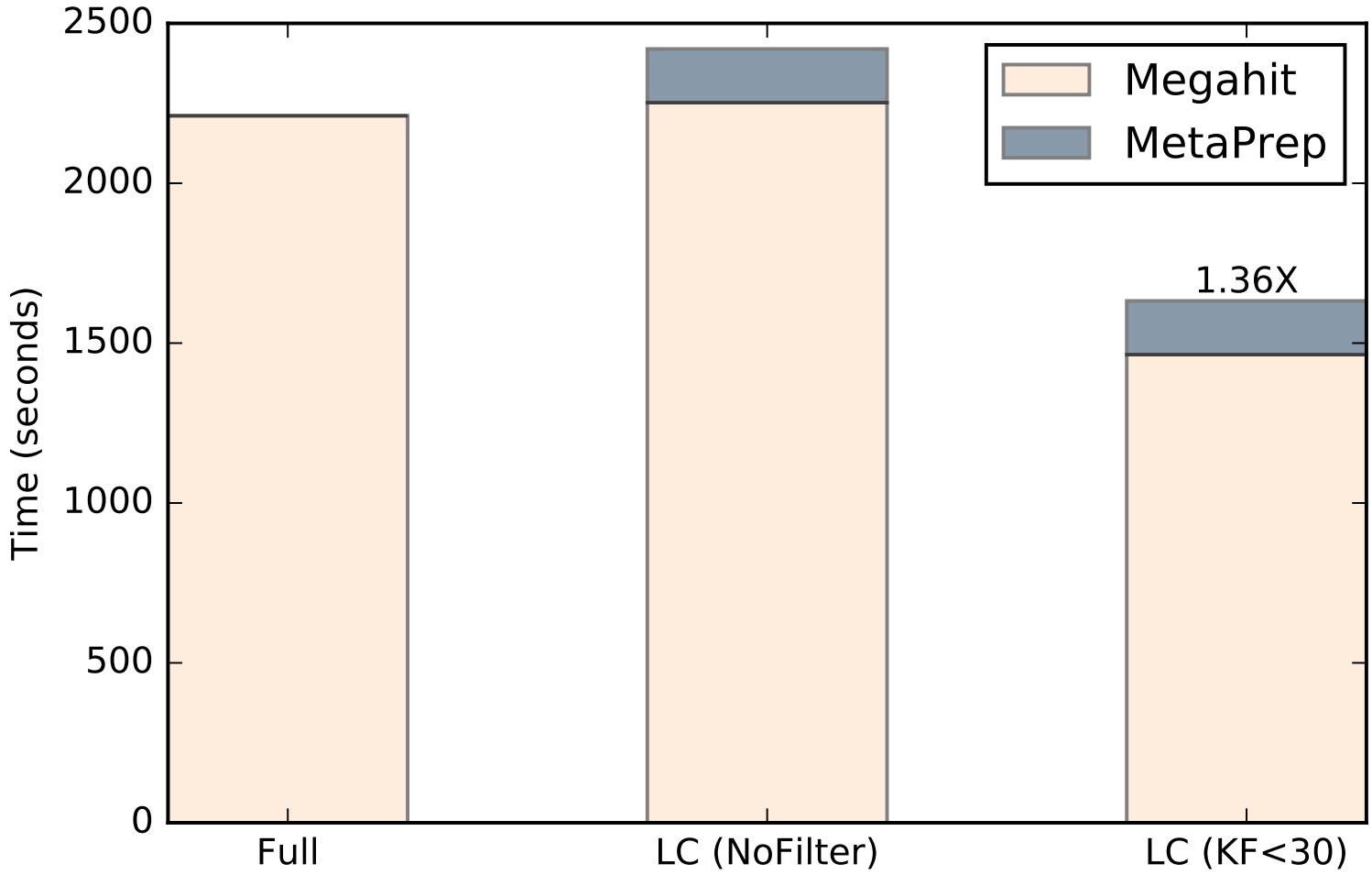
- Largest component size can be reduced by using filters.
  1. *k*-mer size (k) - Longer *k*-mers occur in fewer components
  2. *k*-mer frequency (KF) - Filter erroneous (low frequency) and repeat *k*-mers (high frequency)

Largest component size

- Largest component size can be reduced by using filters.
  1. *k*-mer size (k) - Longer *k*-mers occur in fewer components
  2. *k*-mer frequency (KF) - Filter erroneous (low frequency) and repeat *k*-mers (high frequency)



MEGAHIT single-node execution time for MM dataset



MEGAHIT assembly quality

Table 2: Assembly Quality Comparison - MM dataset.

Type	Contigs	Total (Mbp)	N50 (bp)
No Preproc	24 931	203.65	50 607
No Filter	25 002	203.65	50 550
$KF < 30$	40 632	208.24	23 126

Talk Outline

Motivation for our work

MetaPrep: a new metagenome pre-processing strategy

MetaPrep evaluation
- Parallel Scaling
- Comparison to prior work
- Impact on Metagenome assembly

Conclusions and Future work

Conclusions

1. Developed a new memory efficient parallel workflow for partitioning metagenome dataset into connected components.
2. Speedup up to 4.22$\times$ over AP_LB approach by [Flick2015].
3. We can process a metagenome dataset with 1.13 billion reads (Iowa continuous corn soil) in 14 minutes using 16 nodes of Edison.
4. Preprocessing time (MetaPrep) $\ll$ Assembly time.

Future Work

1. For most datasets, we observe creation of a single large connected component after partitioning the read graph.
   - Splitting components using filters impacts assembly quality.
   - Does scaffolding help in improving assembly quality?
2. Reduce data exchanged in the inter-node communication step of connected components.

Acknowledgment

This research is supported in part by NSF award #1439057. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References I


Sebastian Deorowicz, Marek Kokot, Szymon Grabowski, and Agnieszka Debudaj-Grabysz.
KMC 2: Fast and resource-frugal k-mer counting.
Bioinformatics, 31(10):1569–1576, 2015.


Patrick Flick, Chirag Jain, Tony Pan, and Srinivas Aluru.
A parallel connectivity algorithm for de Bruijn graphs in metagenomic applications.
In *Proc. Int'l. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*, 2015.


Adina Chuang Howe, Janet K. Jansson, Stephanie A. Malfatti, Susannah G. Tringe, James M. Tiedje, and C. Titus Brown.
Tackling soil diversity with the assembly of large, complex metagenomes.
Proceedings of the National Academy of Sciences, 111(13):4904–4909, 2014.

References II

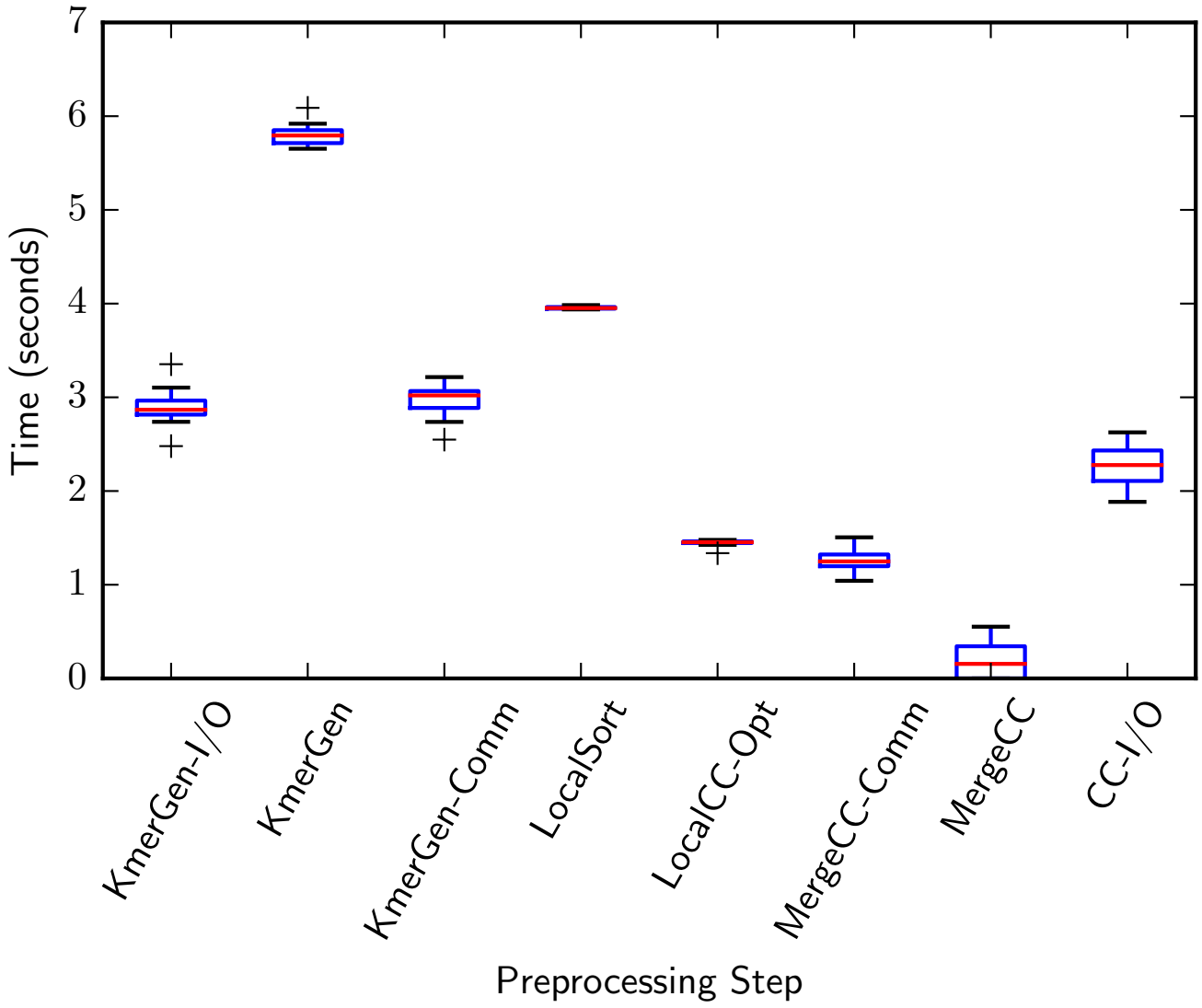
Dinghua Li, Ruibang Luo, Chi-Man Liu, Chi-Ming Leung, Hing-Fung Ting, Kunihiko Sadakane, Hiroshi Yamashita, and Tak-Wah Lam.
MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices.
Methods, 102:3–11, 2016.

Md Mostofa Ali Patwary, Peder Refsnes, and Fredrik Manne.

Multi-core spanning forest algorithms using the disjoint-set data structure.
In *Proc. IEEE Int'l. Parallel & Distributed Processing Symposium (IPDPS)*, 2012.

Thank You

Load Balance among 16 MPI tasks - MM dataset



Multi-pass Execution - MM dataset

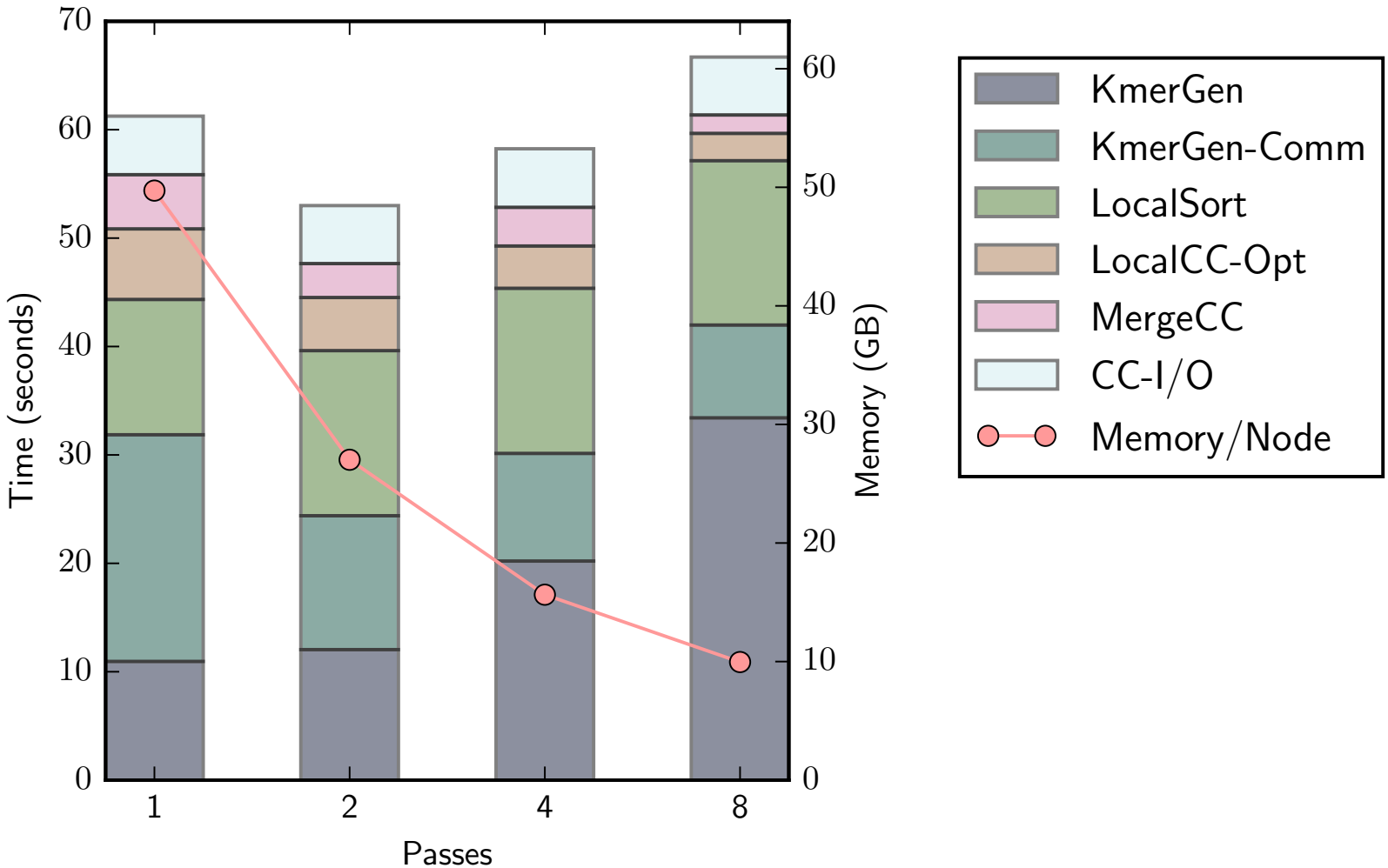


Table 3: Index creation time (sequential).

Dataset	# Chunks	Time (seconds)	
		FASTQPart	merHist
HG	384	32	109
LL	384	32	154
MM	384	33	343
IS	1536	180	5160

Table 4: Impact of k on single-node MetaPrep execution time (MM dataset).

k	Time (seconds)				
	KmerGen	LocalSort	LocalCC-Opt	CC-I/O	Total
27	77.02	55.33	6.41	5.40	144.16
63	59.73	67.60	5.16	5.35	137.84

Table 5: Assembly Quality Comparison.

Dataset	Type	MEGAHIT assembly output statistics			
		Contigs	Total (Mbp)	Max (bp)	N50 (bp)
HG	No Preproc	63 519	116.19	217 183	5071
	No Filter	63 483	116.18	217 183	5098
	LC	58 770	113.83	217 183	5510
	Other	4713	2.35	2860	513
	$KF < 30$	64 571	119.01	217 183	5123
	LC	56 732	110.13	217 183	5687
	Other	7839	8.87	43 863	2271
LL	No Preproc	179 828	165.63	225 770	1273
	No Filter	181 751	166.67	225 805	1263
	LC	141 136	148.75	225 805	1593
	Other	40 615	17.9	4028	432
	$KF < 30$	182 717	168.42	225 770	1275
	LC	140 081	147.51	225 770	1587
	Other	42 636	20.90	43 718	465
MM	No Preproc	24 931	203.65	1 067 762	50 607
	No Filter	25 002	203.65	1 067 762	50 550
	LC	23 959	202.99	1 067 762	50 781
	Other	1043	0.66	5788	695
	$KF < 30$	40 632	208.24	611 608	23 126
	LC	26 233	156.04	611 608	28 135
	Other	14 399	52.19	591 560	12 285