Comparative Analysis of de Bruijn Graph Parallel Genome Assemblers

Carlos Gamboa[1], Esteban Meneses[1]

[1] Costa Rica National High Technology Center
Advanced Computing Laboratory

July 2018

Contents

Introduction

- Next-Generation Sequencing provides vast amount of reads
- Genome sequence assembly: ***Mapping*** and ***De Novo***

Figure: *Tetragonisca angustula*

Figure: *Psidium friedrichsthalianum*

Introduction

- Next-Generation Sequencing provides vast amount of reads
- Genome sequence assembly: ***Mapping*** and ***De Novo***

Figure: *Tetragonisca angustula*

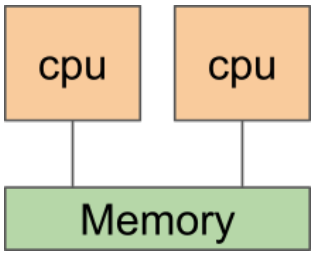
Figure: *Psidium friedrichsthalianum*

Main Objective

Compare and analyze the assemblers to understand them
Regarding, parallel execution time, scalability and sensitivity to the choice of a critical parameter.

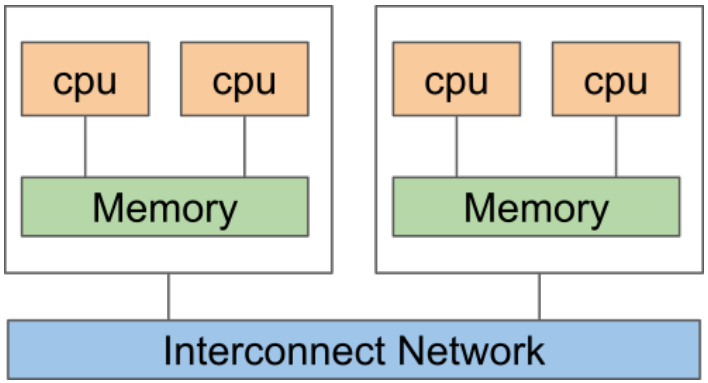
Parallel Programming Paradigms

Shared memory



- OpenMP (Open Multi-Processing)
  Velvet
- Pthreads (POSIX threads)
  SOAPdenovo

Distributed Memory



- MPI (Message Passing Interface)
  ABySS

De Bruijn Graphs

TACGACGTCGACT

$k = 3$

```
TAC
 ACG
  CGA
   GAC
    ACG
     CGT
      GTC
       TCG
        CGA
         GAC
          ACT
```

De Bruijn Graphs

TACGACGTCGACT

$k = 3$

```
TAC
 ACG
  CGA
   GAC
    ACG
     CGT
      GTC
       TCG
        CGA
         GAC
          ACT
```

GACTCCG

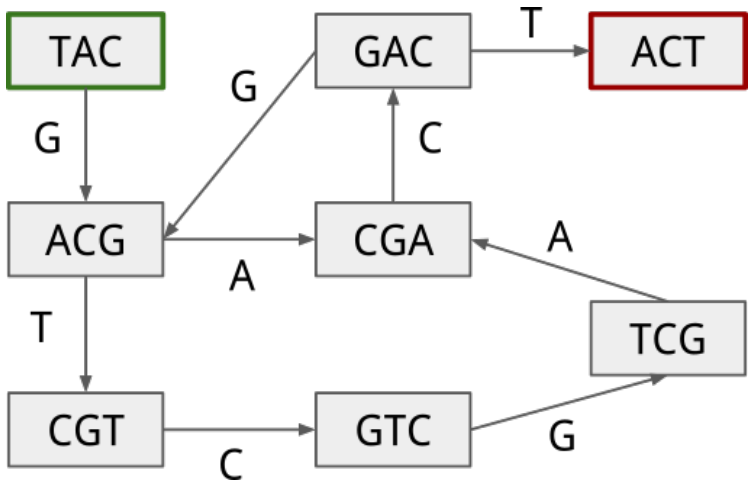


Figure: de Bruijn graph of eight *3-mer* elements

Advantages of de Bruijn graphs

- allows assembly of short reads
- keeping count of k-mers so that repetitive regions are identified.

Drawback

The quality of assembly first rises and then starts to fall with k-mer size approaching the read size.

Experiments and data used

- We did experiments with 1, 2, 4, 8, 16, and 32 cores. Changing the k-mer size. 10 runs per experiment.
- To compare quality of the assembly, parallel execution time, speedup and scalability.

Organism	SRA Accession Number	Genome Size	Total of Reads
E. coli	ERR2213556	5.15 Mb	959K
S. pombe	ERR200231	12.6 Mb	3.3M
S. cerevisiae	ERR1938686	12.2 Mb	3.3M

Table: Data sets used to compare the assemblers

Best quality results for *E. coli* and *S. pombe*

Assembler	N50 (kb)	Contigs	Longest contig (b)	Genome Coverage
Velvet	137889	809	318506	92%
Abyss	125251	590	269444	95%
SOAPdenovo	138475	1204	436364	92%

Table: Assembly results for *E. Coli* data set. Running SOAPdenovo with k=65, Velvet and ABySS with k=31

Assembler	N50 (kb)	Contigs	Longest contig (b)	Genome Coverage
Velvet	73112	2004	334134	97%
Abyss	55918	1532	253768	97%
SOAPdenovo	88850	1503	345419	90%

Table: Assembly results for *S. Pombe* data set, running each assembler with k=35.

Effect of *k* size changes in the parallel execution time

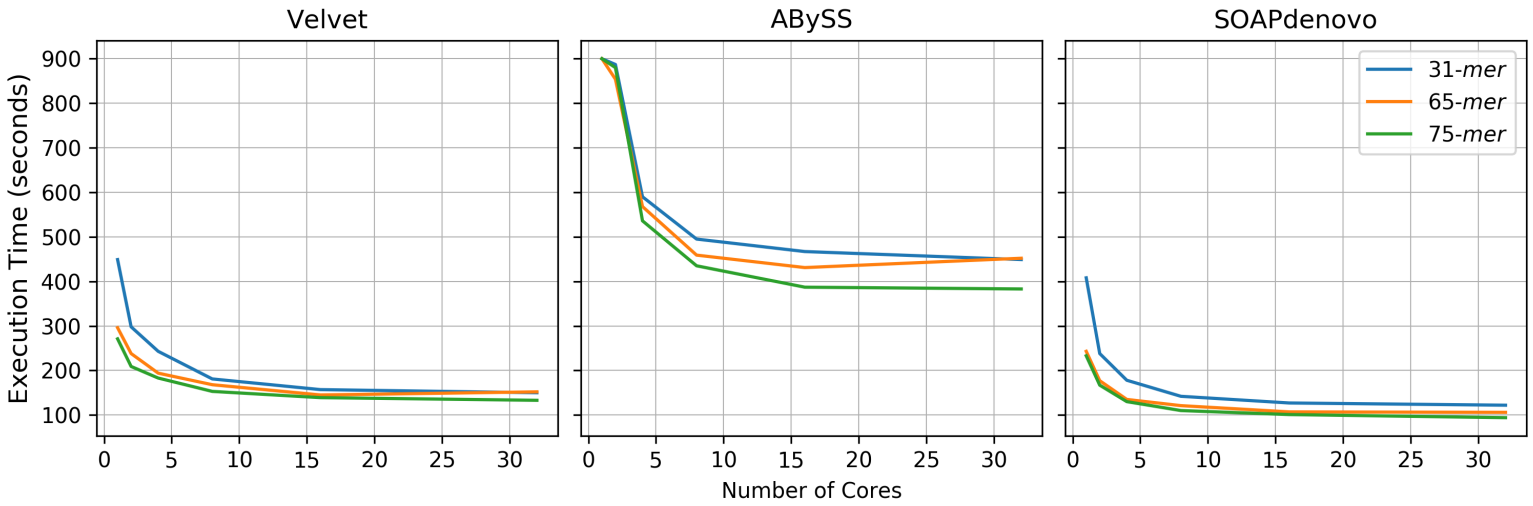


Figure: Average execution times for ***E. coli***

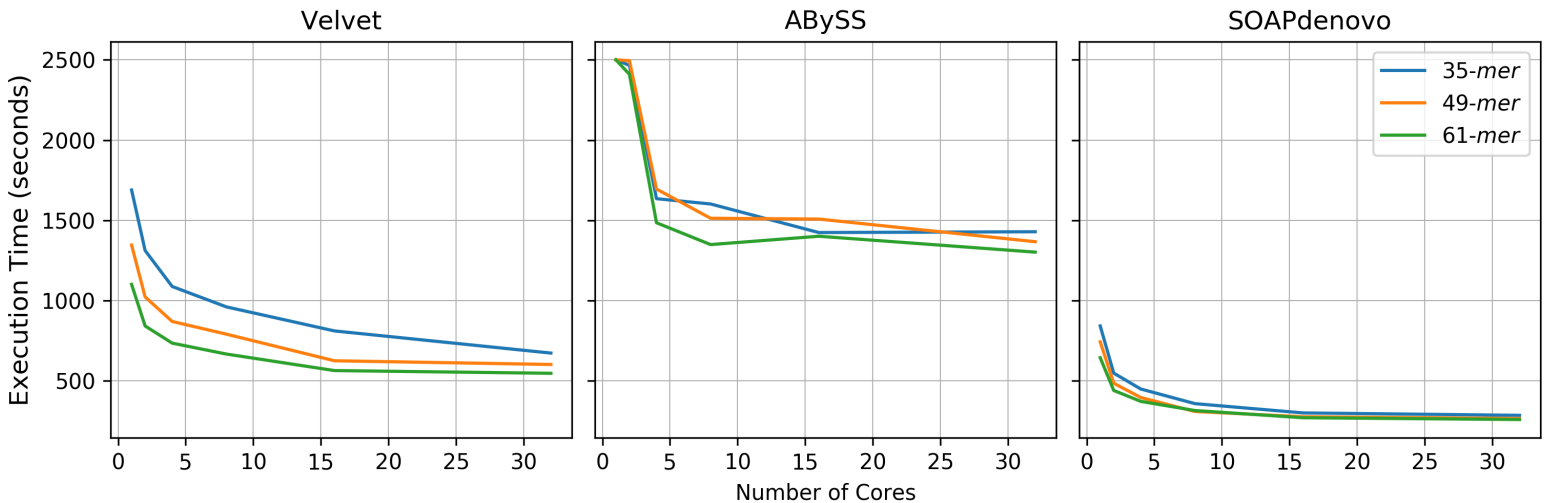


Figure: Average execution times for ***S. pombe***

Non-linear scalability of assemblers with *S. pombe*

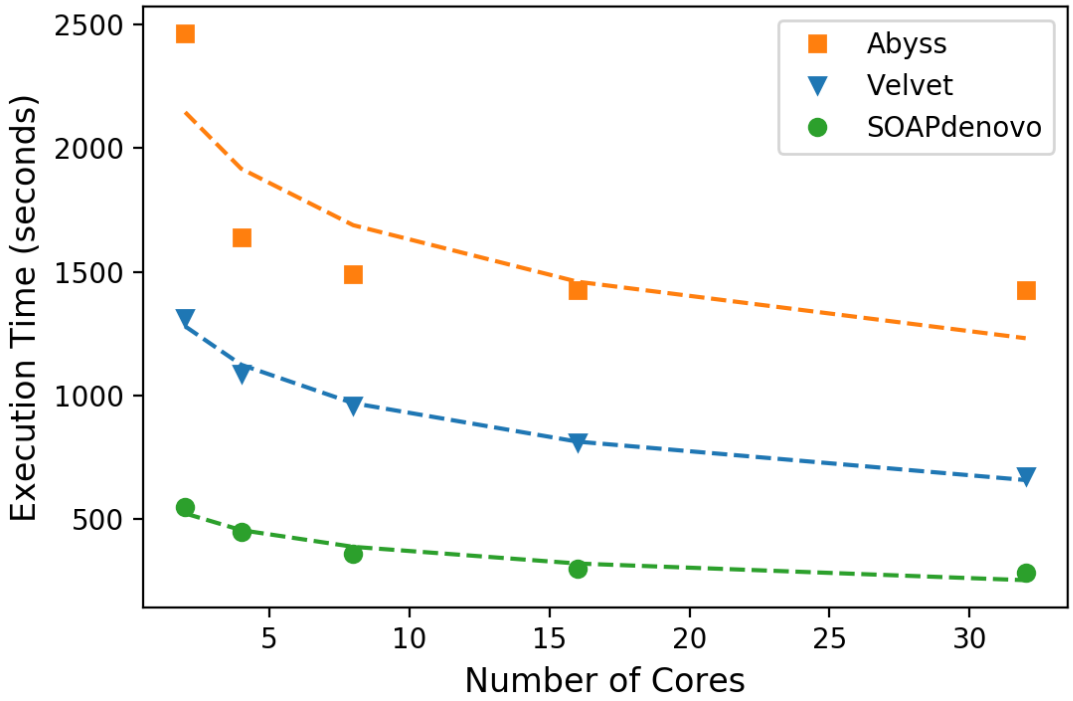


Figure: Parallel execution time of *S. pombe* with k=35 and 16 cores

Performance of assemblers

Data set	Assembler	Max CPU Usage Percentage	Memory Usage (GB)
E. coli	Velvet	97.44 ± 2.3	0.77
	ABySS	100.00 ± 0.01	1.54
	SOAPdenovo	97.71 ± 2.4	4.38
S. pombe	Velvet	95.81 ± 4.0	3.76
	ABySS	100.00 ± 0.01	3.08
	SOAPdenovo	96.88 ± 3.5	5.59

Table: Average memory and average CPU usage of each assembler with ***E. coli*** and ***S. pombe*** with 16 cores.

Performance of assemblers

Data set	Assembler	Max CPU Usage Percentage	Memory Usage (GB)
E. coli	Velvet	97.44 $\pm$ 2.3	0.77
	ABySS	100.00 $\pm$ 0.01	1.54
	SOAPdenovo	97.71 $\pm$ 2.4	4.38
S. pombe	Velvet	95.81 $\pm$ 4.0	3.76
	ABySS	100.00 $\pm$ 0.01	3.08
	SOAPdenovo	96.88 $\pm$ 3.5	5.59

Table: Average memory and average CPU usage of each assembler with ***E. coli*** and ***S. pombe*** with 16 cores.

S. pombe dataset the maximum speedup obtained was 3$\times$ using SOAPdenovo, 2.5$\times$ with Velvet and 1.7$\times$ with ABySS.
S. cerevisiae, SOAPdenovo reached a speedup of 4$\times$ and the other assemblers roughly obtained 2$\times$

Summary of ranges on execution time and quality

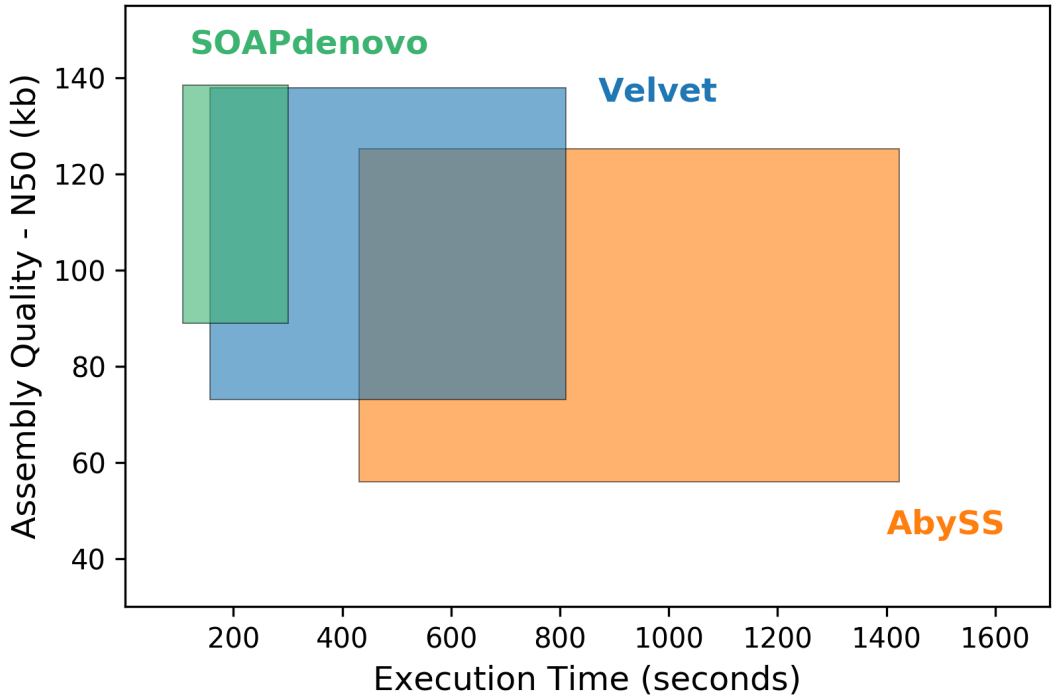


Figure: Ranges of quality and execution time of the assemblers

Conclusions and Future Work

1. Small *k-mer* size affects negatively the execution time of the assemblers
2. Assemblers in the study do not scale linearly
3. SOAPdenovo is the faster assembler. Benefiting of multi-threading paradigm. But, is the least effective in genome coverage
4. ABySS is the assembler with more variability on execution time and quality of the assembly

Conclusions and Future Work

1. Small *k-mer* size affects negatively the execution time of the assemblers
2. Assemblers in the study do not scale linearly
3. SOAPdenovo is the faster assembler. Benefiting of multi-threading paradigm. But, is the least effective in genome coverage
4. ABySS is the assembler with more variability on execution time and quality of the assembly

Future Work

Implementation of a new de novo genome assembler using Parallel Objects

Thanks!

Acknowledgements
Kabré supercomputer at the Costa Rica National High Technology Center

Carlos Gamboa Venegas
cgamboa@cenat.ac.cr