UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Enhancing Transcriptomic Analysis by Influencing De Novo Assembly Using Parallel Computing

Nur Hafizah Zaideen[1], Muhammad Azman Habeeb Mohamed[1], Nor Asilah Wati Abdul Hamid[1,2*], Norazrin Ariffin[3], Mohamed Faris Laham[2] & Zurita Ismail[2]
[1]Faculty of Computer Science and Information Technology
[2]Institute for Mathematical Research
[3]Faculty of Agriculture
Universiti Putra Malaysia
43400 UPM Serdang, Selangor, Malaysia
Email: asila@upm.edu.my

Submitted: 22/8/2024. Revised edition: 9/10/2024. Accepted: 14/10/2024. Published online: 25/11/2024
DOI: https://doi.org/10.11113/ijic.v14n2.483

***Abstract*—The efficient and accurate assembly of genomic data is a computationally intensive process that demands significant computational resources. Traditional sequential approaches often struggle to handle genomic data sets increasing volume and complexity, leading to prolonged execution times and suboptimal results. The study aims to leverage parallel computing capabilities by employing the ABySS and Velvet Assembler tools on the MD2 Pineapple dataset hosted on the Quanta server. By systematically evaluating the performance of these tools across varying thread counts, the study seeks to identify optimal configurations that can enhance the efficiency and accuracy of the *de novo* assembly process, ultimately enabling more rapid and precise genomic analysis. The study found that for the ABySS assembler, an 8-core and 8-thread configuration exhibited the shortest execution time and greatest speedup, while an 8-core and 12-thread setup produced similar outcomes, demonstrating ABySS's flexibility to adjust to various thread configurations. Velvet assembler demonstrated exceptional performance by utilizing 8 cores and 16 threads for the velvetg command, and 8 cores and 8 threads for the velveth command. Significantly, this study provided implications for advancing genomic analysis methodologies by providing valuable guidance on optimizing the efficiency and accuracy of *de novo* assembly processes through careful selection of parallelization configurations, paving the way for future studies and applications in genetic data analysis.**

***Keywords*—Parallel Computing, Quanta Server, Abyss, Velvet, Transcriptome Analysis**

I. INTRODUCTION

In recent years, the field of virology has advanced rapidly, primarily due to increased access to genomic data. Public repositories, such as the National Center for Biotechnology Information (NCBI), now house vast libraries of sequenced genomes, which enables researchers to find new viral genes essential for fighting emerging diseases. Transcriptome assembly plays an important role in understanding the functional components of genetic material in the field of genomics [1]. Efficient computational tools are needed due to the explosion of biological data generated by high-throughput sequencing. *De novo* assembly, the process of constructing transcriptomes or genomes from short DNA sequences, remains a fundamental yet challenging task due to the complexity of gene expression data [2].

High Performance Computing (HPC) provides the computational power necessary to handle these large datasets. HPC systems can accelerate *de novo* assembly by utilising parallel processing, making it possible to efficiently process the increasing volume of genomic data [3-4]. Transcriptomic analysis, a fundamental aspect of genomics, aims to decode an organism's gene expression patterns and understand the dynamics of its transcriptome [5]. *De novo* assembly involves stitching contiguous sequences (contigs) from short DNA fragments obtained through next-generation sequencing [6]. Nevertheless, *de novo* assembly presents significant challenges, including managing sequencing errors, eliminating repetitive regions, and optimizing computational efficiency [7].

Assembly By Short Sequences (ABySS) is a well-known software program that handles *de novo* assembly's complexities. By utilising parallel computing capabilities, ABySS can handle massive transcriptomic datasets more quickly and accurately, which helps to improve assembly speed and accuracy [8]. This allows the assembly algorithm to be computed in parallel over a collection of computers. Developed in C++, ABySS emphasizes its dedication to high-performance parallel computing in transcriptomics research by utilizing the Message Passing Interface (MPI) standard for smooth node communication [9].

The velvet assembler is a popular bioinformatics software program that reassembles the genome from short DNA fragments generated by sequencing technology [10]. These DNA fragments are divided into k-mers, which velvet would split smaller before joining them into long DNA fragments. Velvet assembler is an important tool for expanding our understanding of the development of various organisms because of their adaptability and dependability. By implementing a pipeline for parallel processing *de novo* assembly, incorporating both ABySS and velvet assemblers, we can accelerate the assembly procedure and result in quicker and more precise transcript reconstruction.

Parallel computing is a game-changer in the field of genomics [11]. It boosts the computational capacity to handle large transcriptomic datasets by executing multiple tasks simultaneously across multiple computing resources. Moreover, parallel computing enables concurrent analysis of multiple data subsets to optimize *de novo* assembly for MD2 pineapple transcriptome data, expediting and enhancing the assembly process as shown in [12]. This study leverages two powerful tools, ABySS and Velvet, alongside HPC to enhance the efficiency and precision of transcriptome assembly for the MD2 pineapple dataset. This study aims to provide insights into improving computational efficiency in genomics by optimising configurations of cores and threads.

II. Literature Review

Numerous studies have been conducted to improve the speed and efficiency of transcriptome analysis, which is an important aspect of genomic research. Researchers have focused on improving computational methods to speed up transcriptome analysis procedures as the need for a deeper understanding of gene expression patterns increases. With advancements in high-throughput sequencing technologies and computational algorithms, there has been a concerted effort to develop strategies for accelerating *de novo* assembly, a critical step in unravelling the complexities of gene expression patterns [13]. Nevertheless, despite notable progress in this field, challenges persist, ranging from managing large-scale datasets to optimizing computational efficiency. Hence, this section explores existing literature surrounding efforts to enhance the speed and efficiency of transcriptomic analysis, shedding light on innovative methodologies and their potential implications for advancing genomic research.

A study conducted by [14] suggested that a parallelisation solution for the diBELLA 2D pipeline could improve the performance of *de novo* genome assembly through distributed memory techniques and linear algebra operations. Computationally intensive tasks like matrix multiplication and transitive reduction are parallelized across processors, leading to near-linear scaling and high parallel efficiency. Therefore, this parallelisation strategy promises efficient de novo assembly of large genomes using long reads in distributed memory environments [15]. The Summary of the different parallelisation solutions for advancing genomic research is shown in Table I.

Moreover, the studies conducted by [21] showcase the effectiveness of tools like Trinity, SPAdes, and Trans-ABySS in generating high-quality transcriptome assemblies, providing valuable insights for researchers working on non-model organisms. They demonstrate the species-specific differences in assembly tool performance, emphasizing the need for tailored approaches based on the organism being studied. Furthermore, comparing different assembly strategies, including *de novo* and genome-guided assembly, [22] has shown that *de novo* assembly can be as effective as genome-guided assembly, especially when fragmented reference genomes.

In addition, the temporal progress of gene expression analysis using RNA-Seq data has significantly advanced with the development of computational methods. Various research papers highlight the evolution of computational tools for differential gene expression (DEG) analysis, emphasizing the need for rigorous and efficient methods to explore time-dependent changes in gene expression [24]. RNA-Seq technology has revolutionized transcriptomics research by providing high resolution and a broad dynamic range, leading to the development of numerous computational tools for data analysis, from read preprocessing to DE analysis [25]. Furthermore, integrating new bioinformatics tools for time series experiments in RNA-Seq analysis is discussed, highlighting the continuous improvements and future applications in differential expression analysis [26]. In addition to the prior research, the recent publication by [27-29] enhanced the clustering-based differential expression analysis method for RNA-seq data, providing additional insights and improvements to the bioinformatic tools used in this study area.

III. Methodology

The dataset used in this study was derived from the Illumina platform Next Generation Sequencing (NGS), starting off with RNA extraction and purification from fresh pineapple leaves to the following protocols for creating cDNA libraries. The cDNA libraries were finally sequenced on the Illumina GAIIx platform resulting to six independent RAW files of containing transcriptome data for MD2 pineapple [30]. The methodology chosen for processing the MD2 pineapple transcriptome using parallel computing platforms is the Software Development Life Cycle (SDLC) based on the Agile approach, as shown in Fig. 1.

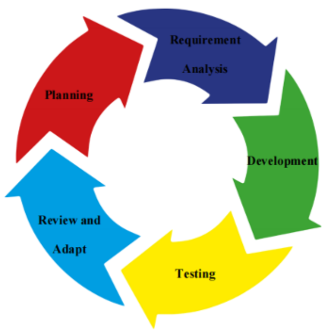

Fig. 1. Agile Methodology

TABLE I. SUMMARY OF THE DIFFERENT PARALLELISATION SOLUTIONS FOR ADVANCING GENOMIC RESEARCH.

Parallelisation Technique	Performance Improvements	Key Features	Tools/Software	Reference
Explicit Multi-Threaded (XMT) assembler.	Speedups between 25x to 61x compared to serial assemblers, though limited by memory.	Parallel processing for large datasets.	XMT Assembler	[19]
Cluster computing for RNA-Seq analysis.	Improved performance on multiple RNA-Seq datasets without requiring bioinformatics expertise.	User-friendly, automated parallel processing.	TRUFA	[20]
Parallel string graph construction and transitive reduction.	Near-linear scaling, high parallel efficiency for large genome assembly.	Distributed memory techniques, MPI for inter-processor communication.	diBELLA 2D	[14]
Distributed-memory technique.	Up to 15x speedup over Hifiasm and 58x over HiCanu for large genome assemblies.	Redistributes sequences, multiway number partitioning.	ELBA	[16]
Multicore processing.	Significant reductions in RNA-Seq processing time with up to four cores.	Multiple core processing, optimized cache speed.	FastQC, Flexbar, Hisat2, Samtools	[17]
Advanced compression for nanopore signal data.	Reduced runtime from two weeks to 10.5 hours for nanopore sequencing.	Optimized data storage and analysis.	SLOW5	[18]
Bayesian approach for transcriptome assembly.	Improved accuracy in gene expression analysis, especially for lowly expressed genes.	Handles alternative splicing, transcript expression probabilities.	BayesDenovo	[23]

Agile methodology is preferred for its adaptability to complex and evolving tasks like transcriptomic studies. By breaking the project into smaller iterations, Agile promotes incremental development and continuous improvement throughout the analysis process. This iterative approach aligns well with the project's multifaceted stages, including planning, data retrieval, assembly, calculation, graph plotting, and comparison. The Agile model's flexibility accommodates changing requirements and emerging insights, which is crucial for optimizing *de novo* assembly and seamlessly integrating parallel processing techniques.

Establishing the study environment using Conda was a crucial step during the planning phase. This setup was instrumental in efficiently managing software dependencies. The process involved downloading the Miniconda installer and initiating the setup via the terminal with a bash command. ABySS, a key component for de novo assembly, was downloaded through Conda, ensuring smooth package management, and resolving dependencies. Using FileZilla for secure data transmission between Windows and Ubuntu platforms and installing Velvet Assembler on the Quanta server further enhanced the project's efficiency.

A. The Requirement Analysis Phase

The requirement analysis phase of Agile is centered around obtaining essential data in our study, the transcriptome data of the MD2 pineapple variety, provided by the Department of Agriculture Technology, Faculty of Agriculture, Universiti Putra Malaysia, played a pivotal role. This FASTQ-formatted dataset was the backbone of our study, serving as crucial material for the subsequent stages. The efficient transfer of this data from a personal Windows machine to the Quanta server using FileZilla ensured prompt and secure access to the required data for further processing. This phase laid a crucial foundation by ensuring the timely and safe availability of necessary data on the selected server.

B. The The Development Phase

The development phase of Agile is dedicated to the de novo assembly process, which is the core task of our project. This process involves the reconstruction of the MD2 pineapple transcriptome using both parallel computing and traditional sequential processing methods. The conversion of short DNA reads into longer sequences, known as contigs, is a key part of this process. Parallel computing techniques, such as dividing sequencing reads into smaller subsets for processing, are employed. The project evaluates performance across different configurations, including varying numbers of cores and threads, alongside sequential processing. The time taken for assembly is recorded for both approaches to provide insights into their speed and performance. This approach allows for the identification of potential areas for enhancement or optimization.

C. The Testing Phase

In the testing phase, the study aims to evaluate and compare the de novo assembly process outcomes, focusing on two key metrics: average execution time and speedup. Average execution time measures the time taken for the assembly process to complete across various setups, providing insights into computational efficiency. Speedup, calculated as the ratio between computation time using a single CPU core and that using multiple cores illustrates the improvement in computational speed achieved through parallel processing. These metrics enable a detailed analysis of performance enhancements resulting from parallelism. Findings from these evaluations inform decisions on optimising future transcriptome analysis, offering crucial insights into the efficiency and effectiveness of parallel computing platforms.

The implementations for the ABySS and Velvet assemblers involve specific commands to optimize their respective De Novo assembly processes. For ABySS, users make repeated changes to the number of cores and threads for each run, testing various combinations to enhance the assembly method. The 'np' function allows customization of the computing workload by specifying the number of processors or cores, maximizing resource usage based on available hardware, while the 'j' command specifies the number of threads to improve parallelization and assembly efficiency. In contrast, the Velvet Assembler uses two successive commands: velveth and velvetg. The velveth command produces the hash table for the input data using the format time velveth <file-name> <k-mer length> -shortPaired -fastq -separate <data-file-1> <data-file-2>, and the velvetg command constructs the de novo assembly with time velvetg <file-name>. This integrated approach ensures both ABySS and Velvet assemblers are effectively customized and utilized according to individual computational resources and requirements.

D. The Review and Adapt Phase

The study conducts a detailed outcomes analysis in the review and adaptation phase by creating graphs for specific core configurations. These graphs illustrate the relationship between the number of threads and execution time, providing information about the assembly process's performance across different configurations. Another set of plots examines the relationship between threads and speed increase for each core configuration, showcasing system scalability and resource optimization. The project remains adaptable to data format or size changes, with the Agile approach facilitating a seamless return to the planning stage if needed. This adaptability ensures that the project can accommodate frequent modifications in data features, which is vital in the dynamic field of bioinformatics research. Agile methodology serves as a solid foundation for responding to evolving needs by allowing adjustments in tool selection and data characteristics.

E. Performance Analysis

To conduct an analysis on a tool's performance, the metrics of speedup and efficiency are chosen. The formulas used to calculate the speedup and efficiency of the tool are as follows:

Speedup is calculated as follows:

$$\text{Speedup} = \frac{\text{Sequential Execution Time}}{\text{Parallel Execution Time}}$$

Efficiency is calculated as follows:

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of Threads}} \times 100$$

These formulas provide quantitative measures to evaluate the performance improvement and resource utilization of the tool when running in parallel compared to sequential execution. A higher speedup value indicates a greater performance gain, while an efficiency value closer to 1 (or 100%) suggests optimal utilization of the available processors.

IV. Results and Discussion

This section provides a comprehensive analysis of the results obtained from de novo assembly procedures, focusing on both the ABySS and Velvet assemblers. In the ABySS segment, we tested various combinations of cores and threads, ranging from 2 to 8 cores and 2 to 16 threads, to gauge their impact on overall performance. In the Velvet section, our aim was to enhance de novo assembly by implementing parallel processing, using a fixed number of 8 cores with different thread counts.

Our initial focus is on the ABySS software, where we meticulously examine the results of each combination over three rounds. The slight variations observed in these rounds could be attributed to system and environmental changes during the process. This underscores the importance of our multi-round approach, which enhances the reliability and comprehensiveness of our analysis.

Table II summarises the outcomes from three rounds of de novo assembly processes, each utilising 2 cores with varying thread counts, which are 2, 4, 8, 12, and 16 threads. Notably, the 2-core and 12-thread configuration stands out for its exceptional performance, boasting an average execution time of 56.67 minutes and the highest speedup recorded at 3.74. This configuration serves as a clear benchmark for the most efficient performance.

TABLE II. FINDINGS WITH A CONSTANT OF 2 CORES

Number of Cores	Number of Threads	Execution Time (minutes)				Speedup
		1st Round	2nd Round	3rd Round	Average	
1	1	212	212	212	212	1
2	2	118	118	117	117.67	1.80
2	4	73	74	73	73.33	2.89
2	8	58	59	57	58	3.66
2	12	60	55	55	56.67	3.74
2	16	86	108	77	90.33	2.35

TABLE III. FINDINGS WITH A CONSTANT OF 4 CORES

Number of Cores	Number of Threads	Execution Time (minutes)				Speedup
		1st Round	2nd Round	3rd Round	Average	
1	1	212	212	212	212	1
4	2	108	108	108	108	1.96
4	4	64	67	64	65	3.26
4	8	48	47	48	47.67	4.45
4	12	48	49	46	47.67	4.45
4	16	67	69	68	68	3.12

Table III shows the outcomes of three rounds of de novo assembly procedures using 4 cores with different thread counts (2, 4, 8, 12, and 16 threads). The results indicate that the best performance is achieved when employing 4 cores with 8 and 12 threads, resulting in an average execution time of 47.67 minutes and a speedup of 4.45. Meanwhile, Table IV summarises outcomes from three rounds of de novo assembly procedures, which use 8 cores with varying thread counts (2, 4, 8, 12, and 16 threads). It highlights the optimal performance with 8 and 12 threads, as evidenced by the shortest average execution time of 42 minutes and the maximum speedup of 5.05.

TABLE IV. FINDINGS WITH A CONSTANT OF 8 CORES

Number of Cores	Number of Threads	Execution Time (minutes)				Speedup
		1st Round	2nd Round	3rd Round	Average	
1	1	212	212	212	212	1
8	2	108	101	101	103.33	2.05
8	4	62	58	58	59.33	3.57
8	8	42	42	42	42	5.05
8	12	48	39	39	42	5.05
8	16	66	62	62	63.33	3.35

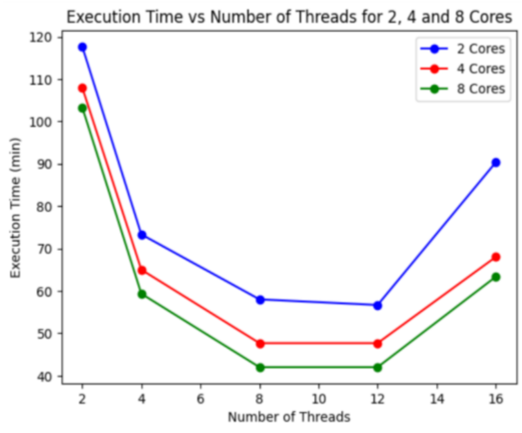

Fig. 2. Number of threads against Execution time with 2, 4 and 8 cores

Fig. 2 illustrates how various combinations of core and thread counts impact the runtime of the de novo assembly process. The graph shows a continuous reduction in execution time as the number of cores rises from 2 to 8. A parallel pattern is observed when analysing the correlation between thread count and execution time. For 2 cores, the execution time decreases from 2 to 8 threads, then increases at thread counts of 12 and 16. For 4 cores and 8 cores, the execution time lowers as the number of threads increases to 8, stabilises at 12 threads (showing similar performance to 8 threads), and then increases again at 16 threads. Once the thread count reaches 16 for all 2, 4, and 8 cores, the execution time starts to increase. This behaviour may indicate a decline in performance due to possible extra costs associated with managing an excessive number of threads.

Next, Fig. 3 displays a detailed analysis of how different combinations of cores and threads affect the speedup in de novo assembly processes. The correlation between core count and speedup demonstrates an apparent enhancement with increased cores, highlighting the system's effective parallelisation and efficient use of computing resources. An analysis of the relationship between thread counts and speedup shows a gradual improvement in speedup as the number of threads increases, reaching a specific limit (e.g., 12 or 16 threads). Once the limit is surpassed, a decrease in performance is observed, indicating reduced advantages from excessive parallelisation.

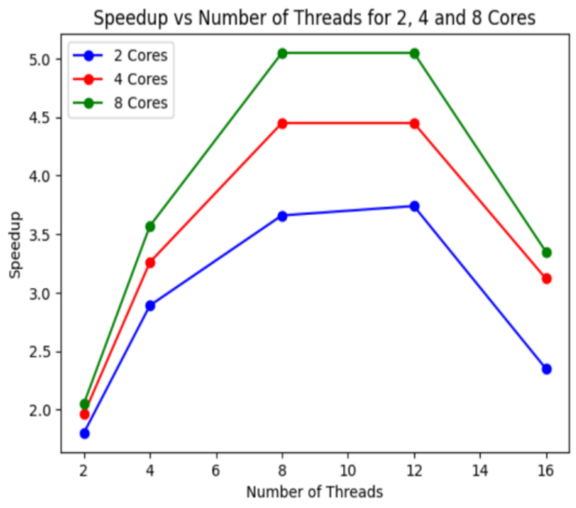

Fig. 3. Number of threads against Speedup with 2, 4 and 8 cores

An improved relationship is shown initially, demonstrating better parallelism as the number of threads increases, highlighting the system's ability to optimise resources for improved performance. However, over a particular limit, adding additional threads can result in a decrease in performance. The main factors driving this issue are contention for shared resources, leading to bottlenecks as several threads compete for a limited number of resources, and a rise in overhead, where handling a higher thread count surpasses the advantages of parallelism. The significant increase in execution time when using 12 or 16 threads indicates that the system has reached a point where the disadvantages of managing more threads outweigh the benefits of parallelisation.

Next, we shall examine the outcomes of the velvet software. Velvet requires two separate commands, velveth and velvetg, for conducting transcriptomic analysis, which sets it apart from ABySS. We will analyse the results of each command individually by changing the number of threads used while keeping a consistent 8-core configuration. Table V displays the performance comparison of the velveth command using different amounts of threads while maintaining a constant of 8 cores. The analysis shows that using 8 cores with 16 threads results in the best performance, with an execution time of 5.03 minutes and a speedup of 3.53.

TABLE V. Result for Time, Speedup, Efficiency for velveth command using 8 Core Quanta Server

Number of Threads	1	2	4	8	16
Time (minutes)	17.77	10.78	6.82	5.53	5.03
Speedup	-	1.65	2.61	3.21	3.53
Efficiency (%)	-	82.5	65.25	40.13	22.06

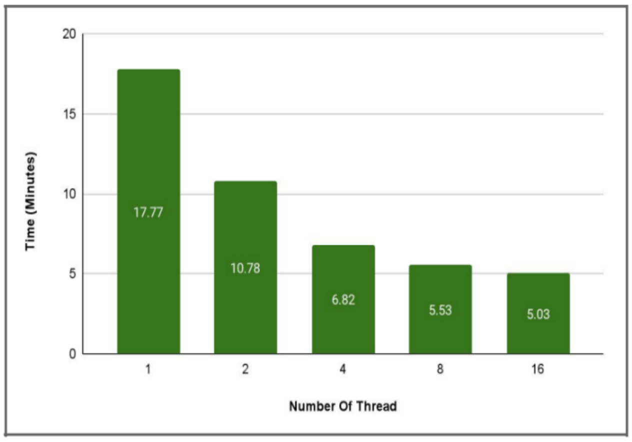

Fig. 4. Bar chart for comparison of Number of Threads against The Execution Time for velveth command using 8 Core Quanta Server

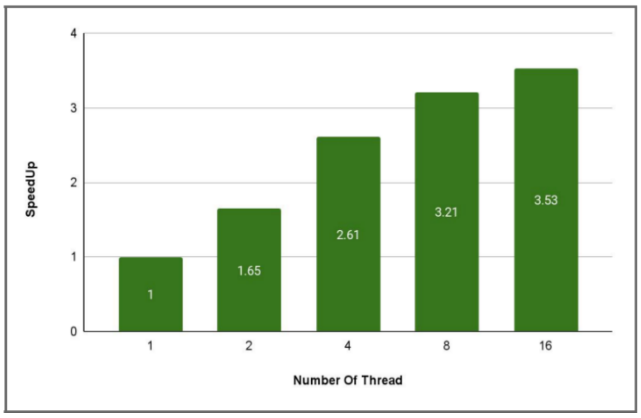

Figure 5. Bar chart for comparison of Number of Threads against The Speedup for velveth command using 8 Core Quanta Server

Two graphs have been generated to visually show the velveth command's performance using different numbers of threads with a fixed 8-core configuration. Fig. 4 displays a bar chart that compares the number of threads with execution time, allowing a clear visual comparison. Fig. 5 illustrates the correlation between the number of threads and speedup, providing valuable information on the efficiency improvements obtained via parallel processing. The graphs show that as the number of threads increases, the execution time decreases, while the speedup of the performance shows a corresponding increase for the velveth command in the *de novo* assembly process.

Following this, Table VI displays the outcomes of executing the velvetg command using different thread counts on a consistent 8-core system. The configuration of 8 cores and 8 threads demonstrates excellent performance, with an execution time of 23.27 minutes and a speedup of 4.37.

TABLE VI. Result for Time, Speedup, Efficiency for velvetg command using 8 Core Quanta Server

Number of Threads	1	2	4	8	16
Time (minutes)	101.68	57.41	34.67	23.27	25.26
Speedup	-	1.77	2.93	4.37	4.03
Efficiency (%)	-	88.5	73.25	54.63	25.19

The results are also displayed graphically to offer a more detailed viewpoint. Fig. 6 shows a bar chart comparing thread count with execution time for the velvetg command using 8 cores. Meanwhile, Fig. 7 displays a bar chart comparing thread count with speedup for the same setup on a Core Quanta Server.

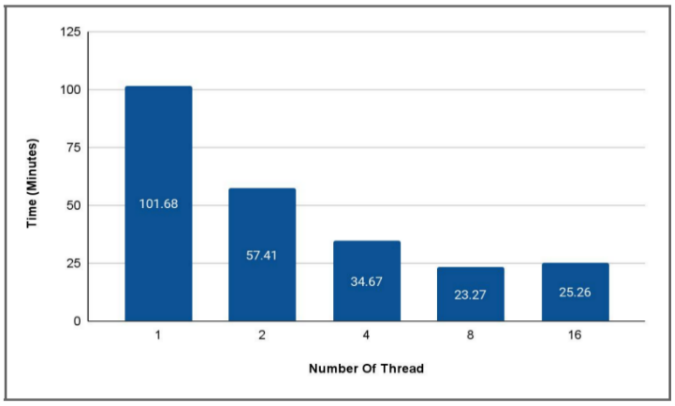

Fig. 6. Bar chart for comparison of Number of Threads against The Execution Time for velvetg command using 8 Core Quanta

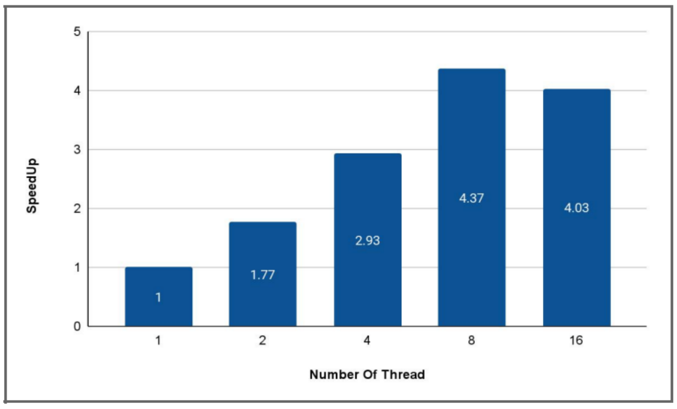

Fig. 7. Bar chart for comparison of Number of Threads against The Speedup for velvetg command using 8 Core Quanta

Analysed from the graphs of the velvetg command, it is evident that an increase in the thread count leads to a reduction in the time taken for execution. However, velvetg's execution time is longer with 16 threads than 8 threads, and the speedup is smaller with 16 threads than 8 threads. There may be an imbalance between the advantages of parallelism and the additional overhead of using 16 threads. Thread overhead and resource contention are probable causes of inefficiencies and bottlenecks in the assembly process when using a higher number of threads.

Moreover, Fig. 8 provides a bar chart illustrating the efficiency of both velveth and velvetg commands across the usage of different numbers of threads. When utilising multiple threads, we often observed that velvetg performs more efficiently than velveth in our de novo assembly project. Velvetg consistently outperformed velveth in optimising the assembly across various thread counts. In our study on using Velvet, parallelisation benefits are more evident during the assembly optimisation phase with velvetg than during the initial hash building with velveth.

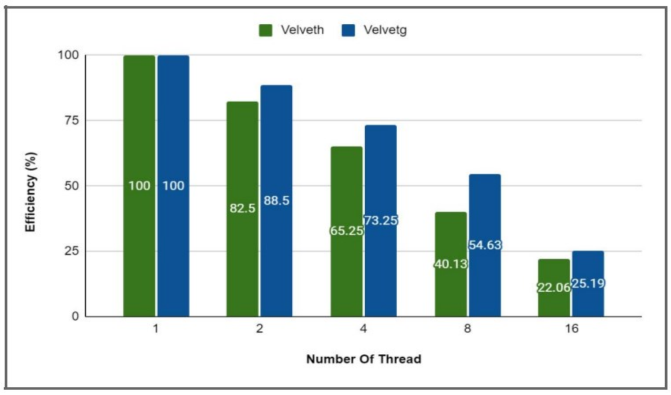

Fig. 8. Bar chart for comparison between velveth and velvetg command efficiency across the usage of different number of threads

V. Conclusions

In conclusion, our investigation into *de novo* assembly processes using ABySS and Velvet assemblers has yielded significant findings. ABySS demonstrated its superiority with an 8-core and 8-thread setup while proving its adaptability with an 8-core and 12-thread configuration. Conversely, Velvet showcased exceptional performance with 8 cores and 16 threads for the velveth command and 8 cores and 8 threads for the velvetg command. These results underscore the importance of careful configuration selection for optimal performance in de novo assembly procedures.

Our study not only deepens our understanding of the relationships among core count, thread count, execution time, and speedup for ABySS and Velvet assemblers but also provides practical guidance for researchers and practitioners. The diminishing returns observed with higher thread counts underscore the need for a balanced approach to parallelisation. By carefully selecting configurations, researchers can optimise the performance of *de novo* assembly procedures. These findings are particularly valuable for those working in parallelised computing environments and aiming to enhance the efficiency of transcriptome analysis using ABySS and velvet assemblers.

It is important to acknowledge the limitations of our study. One notable constraint is the absence of GPU utilization in the ABySS and Velvet Assembler program, which could have significantly improved processing efficiency. Relying solely on CPU resources may have restricted the speed and scalability of the *de novo* assembly operations. These limitations underscore the need for future research to focus on integrating modern hardware resources, particularly GPUs, to maximize efficiency in transcriptomic analysis. The observed restrictions also highlight the ongoing need for assemblers to adapt and integrate with advanced hardware resources to optimize the efficiency of transcriptomic analysis.

Acknowledgment

This research was supported by Faculty of Computer Sciences and Information Technology, and Institute for Mathematical Research, Universiti Putra Malaysia.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

References

[1] I. Boria, L. Boatti, G. Pesole, and F. Mignone. (2013). NGS-trex: next generation sequencing transcriptome profile explorer. *BMC bioinformatics*, *14*, 1-8.

[2] M. Ellis, E. Georganas, R. Egan, S. Hofmeyr, A. Buluç, B. Cook, L. Oliker, & K. Yelick. (2017). Performance characterization of de novo genome assembly on leading parallel systems. *In Euro-Par 2017: Parallel Processing: 23rd International Conference on Parallel and Distributed Computing, Santiago de Compostela, Spain, August 28–September 1, 2017, Proceedings 23* (pp. 79-91). Springer International Publishing.

[3] H. Peréz-Sánchez, J. M. Cecilia, and I. Merelli. (2014). The role of high performance computing in bioinformatics. 2nd International Work-Conference on Bioinformatics And Biomedical Engineering. *Proceedings of 2nd International Work-Conference on Bioinformatics and Biomedical Engineering*.

[4] U. Aldasoro, L. F. Escudero, M. Merino, and G. Perez. (2017). A parallel branch-and-fix coordination based matheuristic algorithm for solving large sized multistage stochastic mixed 0–1 problems. *European Journal of Operational Research*, *258*(2), 590-606.

[5] A. Sudhagar, G. Kumar, and M. El-Matbouli. (2018). Transcriptome analysis based on RNA-Seq in understanding pathogenic mechanisms of diseases and the immune system of fish: a comprehensive review. *International Journal of Molecular Sciences*, *19*(1), 245.

[6] E. Georganas, R. Egan, S. Hofmeyr, E. Goltsman, B. Arndt, A. Tritt, A. Buluç, L. Oliker and K. Yelick (2018). Extreme scale de novo metagenome assembly. *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 122-134). IEEE.

[7] X. Liao, M. Li, Y. Zou, F. Wu, and J. Wang. (2019). Current challenges and solutions of de novo assembly. *Quantitative Biology*, *7*(2), 90-109.

[8] I. Birol, S. D. Jackman, C. B. Nielsen, J. Q. Qian, R. Varhol, G. Stazyk, R. D. Morin, Y. Zhao, M. Hirst, J. E. Schein, and D. E. Horsman. (2009). De novo transcriptome assembly with ABySS. *Bioinformatics*, *25*(21), 2872-2877.

[9] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. (2009). ABySS: a parallel assembler for short read sequence data. *Genome research*, *19*(6), 1117-1123.

[10] E. Costa and G. Silva, (2023). The velvet assembler using OpenACC directives. *Proceedings of International Conference on Bioinfor*. *92*, 72-81.

[11] O. Gupta, S. Rani, and D. C. Pant. (2011). Impact of parallel computing on bioinformatics algorithms. *Proceedings 5th IEEE International Conference on Advanced Computing and Communication Technologies*. 206-209.

[12] W. D. Ong, L.-Y. C. Voo, and V. S. Kumar. (2012). De novo assembly, characterization and functional annotation of pineapple fruit transcriptome through massively parallel sequencing.

[13] B. G. Jackson, M. Regennitter, X. Yang, P. S. Schnable, and S. Aluru. (2010). Parallel de novo assembly of large genomes from high-throughput short reads. *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)* (pp. 1-10). IEEE.

[14] G. Guidi, O. Selvitopi, M. Ellis, L. Oliker, K. Yelick, and A. Buluç. (2021). Parallel string graph construction and transitive reduction for de novo genome assembly. *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (pp. 517-526). IEEE.

[15] S. B. Iryanto, W. A. Kusuma, and H. Sukoco, (2018). Performance analysis of parallel de novo genome assembly in shared memory system. *IOP Conference Series: Earth and Environmental Science* (Vol. 187, No. 1, p. 012032). IOP Publishing.

[16] G. Guidi, G. Raulet, D. Rokhsar, L. Oliker, K. Yelick, and A. Buluç. (2022). Distributed-memory parallel contig generation for de novo long-read genome assembly. *Proceedings of the 51st International Conference on Parallel Processing* (pp. 1-11).

[17] L. J. Bin, N. A. W. A. Hamid, Z. Ismail, and M. F. Laham. (2021). Fast processing rna-seq on multicore processor. *Baghdad Science Journal*, *18*(4 Suppl.), 1413-1413.

[18] H. Gamaarachchi, H. Samarakoon, S. P. Jenner, J. M. Ferguson, T. G. Amos, J. M. Hammond, H. Saadat, M. A. Smith, S. Parameswaran, and I.W. Deveson. (2022). Fast nanopore sequencing data analysis with SLOW5. *Nature biotechnology*, *40*(7), pp.1026-1029.

[19] D. Naishlos, J. Nuzman, C.-W. Tseng, and U. Vishkin. (2001). Evaluating the XMT parallel programming model. In *International Workshop on High-Level Parallel Programming Models and Supportive Environments* (pp. 95-108). Berlin, Heidelberg: Springer Berlin Heidelberg.

[20] E. Kornobis, L. Cabellos, F. Aguilar, C. Frías-López, J. Rozas, J. Marco, and R. Zardoya, 2015. TRUFA: a user-friendly web server for de novo RNA-seq analysis using cluster computing. *Evolutionary Bioinformatics*, *11*, EBO-S23873.

[21] M. Hölzer and M. Marz. (2019). De novo transcriptome assembly: A comprehensive cross-species comparison of short-read RNA-Seq assemblers. *Gigascience*, *8*(5), giz039.

[22] P. Carvajal-Lopez, F. D. von Borstel, A. Torres, G. Rustici, J. Gutierrez, and E. Romero-Vivas. (2018). Microarray-based quality assessment as a supporting criterion for de novo transcriptome assembly selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *17*(1), 198-206

[23] X. Shi, X. Wang, A. F. Neuwald, L. Halakivi-Clarke, R. Clarke, and J. Xuan. (2021). A Bayesian approach for accurate de novo transcriptome assembly. *Scientific reports*, *11*(1), 17663.

[24] J. Costa-Silva, D. S. Domingues, D. Menotti, M. Hungria, and F. M. Lopes. (2023). Temporal progress of gene expression analysis with RNA-Seq data: A review on the relationship between computational methods. *Computational and Structural Biotechnology Journal*, *21*, 86-98.

[25] J. Costa-Silva, D. S. Domingues, D. Menotti, M. Hungria, and F. M. Lopes. (2021). Computational methods for differentially expressed gene analysis from RNA-Seq: an overview. *arXiv preprint arXiv:2109.03625*.

[26] S. Oh, S. Song, G. Grabowski, H. Zhao, and J. P. Noonan. (2013). Time series expression analyses using RNA-seq: A statistical approach. *BioMed Research International*, *2013*(1), 203681.

[27] D. Rosati, M. Palmieri, G. Brunelli, A. Morrione, F. Iannelli, E. Frullanti, and A. Giordano. (2024). Differential gene expression analysis pipelines and bioinformatic tools for the identification of specific biomarkers: A review. *Computational and Structural Biotechnology Journal*.

[28] M. Makino, K. Shimizu, and K. Kadota. (2024). Enhanced clustering-based differential expression analysis method for RNA-seq data. *MethodsX*, *12*, 102518.

[29] L. A. E. Nagai, S. Lee, and R. Nakato. (2024). Protocol for identifying differentially expressed genes using the RumBall RNA-seq analysis platform. *STAR Protocols*, *5*(1), 102926.

[30] N. Ariffin, D. W. Newman, M. G. Nelson, R. O'cualain, and S. J. Hubbard. (2024). Proteogenomic Gene Structure Validation in the Pineapple Genome. *Journal of Proteome Research*, *23*(5), 1583-1592.