Vol. 8, No. 3, June 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Empirical CPU–Memory Benchmarking for Long-Read Genome Assembly Resource Optimization in High-Performance Computing

Fatayat[1], Tisha Melia[2], Dwipa Amedihardjo[3]

[1,2,3]Department of Information Systems, Faculty of Computer Science, Riau University, Pekanbaru, Indonesia

Received:
October 10, 2025
Revised:
April 19, 2026
Accepted:
May 27, 2026
Published:
June 22, 2026

Corresponding Author:
Author Name*:
Fatayat
Email*:
fatayat@lecturer.unri.ac.id

DOI:
10.63158/journalisi.v8i3.1602

Abstract. Efficient resource utilization is a critical challenge in High-Performance Computing (HPC) environments, particularly for long-read genome assembly workflows that require substantial computational resources. This study presents an empirical benchmarking framework to optimize resource allocation for de novo long-read genome assembly of Acacia crassicarpa. Nine experimental scenarios were evaluated by varying CPU cores (32, 48, and 64) and memory allocations (32 GB, 64 GB, and 128 GB) managed via the Slurm workload manager. Performance was assessed based on execution time, assembly continuity (N50), and biological completeness using BUSCO. The results demonstrate that CPU scalability significantly impacts performance, reducing execution time by up to 49% when scaling from 32 to 64 cores. Conversely, increasing memory allocation beyond 64 GB yielded no significant improvements in assembly quality, highlighting the risks of resource over-provisioning. Scenario 2 (64 CPU cores and 64 GB RAM) was selected as the optimal configuration because it balanced runtime, N50 continuity, memory efficiency, and BUSCO completeness, not because it produced the absolute shortest runtime. Under Scenario 2, the workflow achieved an average runtime of 59 hours 39 minutes 40 seconds, an N50 value of 7.8 Mb, and a genome completeness score of 99.8%. These findings provide practical guidance for resource planning and workload scheduling in shared HPC-based genomic workflows.

Keywords: High Performance Computing; Slurm workload manager; long-read assembly benchmarking; BUSCO-based completeness assessment; resource optimization.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

High Performance Computing (HPC) has fundamentally established itself as the primary backbone for accelerating scientific discovery in the modern era. The integration of massive parallel architectures, scalable workflow systems, and optimized computational pipelines within HPC ecosystems has enabled breakthroughs across data-intensive computing and bioinformatics [1], [2], [3], [4], [5] In the context of genomics, the exponential growth of sequencing data driven by Third-Generation Sequencing (TGS) technologies has created computational demands that exceed the capabilities of conventional workstations [6], [7], [8]. Consequently, shared HPC infrastructures have become essential for large-scale genome analysis workflows that require substantial computational and memory resources [6], [9], [10]. In Indonesia, the availability of advanced computing hardware alone does not guarantee efficient utilization; without appropriate resource management strategies, HPC systems may experience low utilization efficiency and reduced throughput [3], [11].

A persistent challenge in shared HPC environments is the absence of empirically validated resource allocation strategies. In practice, researchers must manually specify Central Processing Unit (CPU) cores and Random Access Memory (RAM) requests through workload managers such as Slurm before execution. This often leads to a phenomenon known as over-provisioning, where users request resources significantly larger than the actual runtime requirements of their applications [12], [13], [14], [15]. Over-provisioning is typically motivated by attempts to avoid Out-of-Memory (OOM) failures or by the assumption that larger allocations inherently improve performance. However, prior studies show that many scientific and bioinformatics workloads are memory-bound rather than CPU-bound, meaning performance is constrained more by memory bandwidth and access patterns than by additional processing cores [12], [16], [17]. Inefficient allocation creates stranded resources, where allocated memory or CPU capacity remains idle while preventing other users from accessing the cluster, ultimately increasing queue waiting time and reducing overall system efficiency [10], [13], [14]. Recent research on runtime prediction and adaptive resource scheduling further emphasizes that accurate estimation of workload behavior is critical for improving cluster utilization,

yet practical benchmarking studies in real production environments remain limited [3], [4], [12].

In genomics, long-read sequencing technologies such as Oxford Nanopore Technologies (ONT) have enabled the reconstruction of complex genomes by producing reads capable of spanning repetitive regions that are difficult to resolve using short-read data [6], [18]. Nevertheless, long-read data introduces significant computational challenges due to high error rates and large data volumes, requiring intensive computation during correction, alignment, and assembly stages [19]. Genome assembly workflows therefore represent highly demanding HPC workloads, often involving multiple computational phases with heterogeneous scaling behavior [9], [20]. While recent research has explored algorithmic acceleration through GPU and FPGA implementations [16], [17], these approaches frequently depend on specialized hardware that may not be available in many institutional HPC clusters, where CPU-based execution remains the dominant paradigm. As a result, optimizing CPU and memory allocation within existing HPC environments remains a practical and relevant research direction.

To address this challenge, this study focuses on performance analysis using a real-world high-complexity workload: de novo genome assembly of *Acacia crassicarpa* [21], [22]. This species represents an economically important forestry commodity in tropical regions and is widely used in the pulp and paper industry. Despite its industrial significance, genomic resources for *A. crassicarpa* remain limited, making efficient genome assembly an important computational task. The assembly process utilizes long-read sequencing data and the NextDenovo assembler [23], which is known for its performance-oriented algorithms but also for its substantial computational requirements. Such characteristics make it an ideal case study for evaluating resource allocation behavior in HPC environments.

The primary contribution of this research lies in addressing the research gap between theoretical runtime modeling and practical resource management through a systematic investigation of CPU scalability and RAM allocation. Unlike previous studies that focus on hardware acceleration techniques (GPU/FPGA) or predictive algorithms, this work provides a novel empirical benchmarking framework within a real Slurm-managed HPC

environment to identify optimal configurations that mitigate over-provisioning. By evaluating computational efficiency alongside biological assembly quality metrics, such as N50 and BUSCO completeness, this study offers a validated reference for long-read genomics workflows. Consequently, these findings serve as an actionable, practical guide for genomics researchers to strategically baseline their pipeline computational requirements and for HPC administrators to optimize workload scheduling policies, thereby directly enhancing the multi-user sustainability, resource utilization efficiency, and overall throughput of shared institutional HPC resources.

2. METHODS

This research was conducted using a systematic workflow consisting of data collection, computational environment setup, data preprocessing, genome assembly experiments, and final result evaluation, as illustrated in Figure 1. The proposed empirical benchmarking framework is organized into five distinct sequential subsections to ensure experimental transparency and reproducibility.

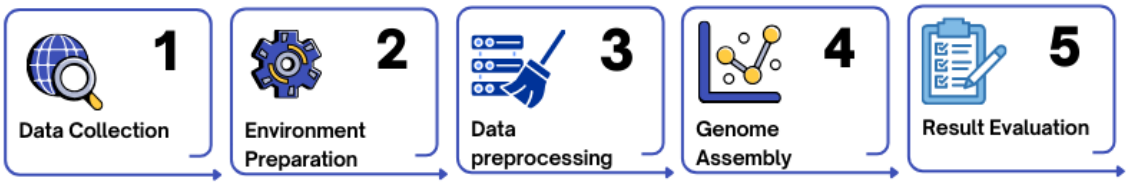

Figure 1. Research Methodology

2.1 Data Collection

The primary dataset used in this study was the raw genome sequence of *A. crassicarpa*, generated using the ONT platform [18]. Long-read sequencing technology was selected because it produces extended read lengths that are essential for resolving complex and repetitive genomic regions commonly found in Acacia genomes. Although ONT data typically exhibits higher error rates compared to short-read sequencing technologies, it provides superior structural information that improves genome assembly continuity [24]. The raw sequencing data was obtained in compressed FASTQ format with a total size of approximately 55.2 GB, serving as the high-complexity workload for all subsequent benchmark treatments.

2.2 Environment Preparation

To accommodate the high computational demands of this dataset, all experiments were executed on the Mahameru High-Performance Computing (HPC) system managed by BRIN. The experiments utilized a General Purpose Compute Node equipped with dual-socket AMD EPYC™ 7513 "Milan" processors, providing 64 physical cores (128 threads) and 256 GB DDR4 ECC RAM. This configuration is suitable for parallel processing workloads [25], which are commonly found in genome assembly tasks [24], [26]. To ensure software compatibility and prevent dependency conflicts with the host system, an isolated virtual environment was created using Conda.

2.3 Data Preprocessing

Within this environment, an initial quality control (QC) step was performed using NanoPlot [27] to evaluate read length distributions and quality score profiles prior to assembly. This preprocessing stage processed the compressed 55.2 GB FASTQ raw dataset to resolve the exact empirical profile of the input reads required to analyze downstream computational constraints. The QC output revealed a highly heterogeneous read-length distribution where the majority of sequences fell within the 1,000 to 30,000 base range, exhibiting a clear distribution peak concentrated at approximately 8,000 bases. Furthermore, the phred quality score profiling indicated that the majority of long reads were distributed within a strict baseline range of Q15 to Q20, which translates to an estimated base accuracy of 93% to 99%. Documenting these baseline data matrices is critical since the high volume of noisy alignments directly dictates the CPU thread consumption and RAM boundaries required during the subsequent error-correction and graph-construction phases.

2.4 Genome Assembly Experiments

Following preprocessing, genome assembly was performed using NextDenovo, a long-read assembler designed to process noisy sequencing data using a Correction-then-Assemble (CTA) strategy [23]. This strategy separates the workflow into two major phases: a CPU-intensive read correction stage and a memory-intensive genome graph construction stage. The NextDenovo parameters tuned in this study are presented in Table 1, where alternative values for each variable are shown in curly braces. All other

parameters were retained at their default settings. The experimental design consisted of nine benchmark scenarios executed under the SLURM workload manager (Table 2). Resource allocation was systematically varied by combining three CPU configurations (32, 48, and 64 cores) with three memory limits (32 GB, 64 GB, and 128 GB). To ensure high statistical power, each of the nine distinct benchmark scenarios was repeated exactly three times independently, yielding a balanced dataset of 27 comprehensive runtime observations for the subsequent Analysis of Variance (ANOVA).

Table 1. Parameter used for NextDenovo

Parameter	Configuration	Description
sort_options	-m {16G, 32G, 64G, 128G}	Memory limit for read sorting process
minimap2_options_raw	-t {32, 48, 64}	Number of threads for read alignment during the read correction stage
minimap2_options_k	-t {32, 48, 64}	Number of threads for read alignment during the assembly stage

Table 2. Scenario used for Benchmarking

Variable	Core	RAM
1	64	128
2	64	64
3	64	32
4	48	128
5	48	64
6	48	32
7	32	128
8	32	64
9	32	32

To eliminate computational resource leakage and guarantee benchmarking integrity, SLURM resource requests via #SBATCH directives were rigidly synchronized with the internal NextDenovo runtime configuration file (run.cfg). Specifically, the multi-threading allocation (minimap2_options) and the core memory sorting parameters (sort_options)

were dynamically matched to mirror the exact hardware constraints requested from the Slurm scheduler. For example, a target run with 64 cores and 64 GB RAM was executed by locking the cluster job via #SBATCH --cpus-per-task=64 and #SBATCH --mem=64G, while explicitly setting minimap2_options_raw=-t 64 and sort_options=-m 64G inside the configuration script. This alignment ensured that the assembler operated strictly within the assigned hardware limits, enabling consistent performance comparisons across scenarios. This strict technical synchronization ensures that resource leakage is eliminated, as the assembler is constrained to operate precisely within the hardware boundaries defined by the SLURM scheduler, thereby maintaining the integrity of the benchmarking process.

2.5 Evaluation and Statistical Framework

The experimental results were evaluated using three key dimensions: computational efficiency, structural continuity, and biological accuracy. Computational efficiency was measured using the total wall-clock execution time recorded by the SLURM scheduler. Structural continuity was assessed using the N50 metric, which reflects the continuity of assembled genome fragments [28]. To ensure that resource reduction did not compromise biological quality, assembly completeness was evaluated using BUSCO against the embryophyta_odb10 lineage dataset containing 1,614 conserved plant genes [29]. This evaluation allowed verification that essential genomic information remained intact across different resource configurations.

To quantitatively evaluate the effects of computational resources on runtime performance, a two-way Analysis of Variance (ANOVA) was conducted using the R statistical environment. The analysis considered CPU core count and memory allocation (as fixed factors, while total runtime served as the response variable. The ANOVA model was designed to assess (i) the independent effect of CPU cores, (ii) the independent effect of memory capacity, and (iii) the interaction effect between CPU and memory configurations. Prior to analysis, runtime values were standardized into numerical hour units to ensure consistent comparison across experiments. Statistical significance was evaluated at a confidence level of 95% ($\alpha = 0.05$). This analysis enabled objective identification of the primary factors influencing computational performance and supported the selection of the optimal resource configuration. To maintain strict

statistical rigor, the ANOVA model's assumptions were thoroughly validated prior to extracting final statistics. First, a Shapiro-Wilk test and residual plotting were applied for the residual normality check. Second, Levene's test was executed as the homogeneity-of-variance check. These diagnostic mathematical procedures ensure that the resulting p-values and F-statistics provide an un-biased, highly accurate interpretation of the cluster dynamics.

3. RESULTS AND DISCUSSION

3.1. Quality Control Results and Computational Implications

Prior to performance benchmarking, the raw ONT sequencing data was evaluated to ensure that the dataset is of good quality. The quality control analysis performed using NanoPlot revealed a highly heterogeneous read-length distribution, which is characteristic of ONT sequencing data (Fig. 2). The majority of reads falls between 1,000 to 30,000 bases in length, providing strong long-range structural information required to resolve repetitive genomic regions. The peak distribution for reads lie in 8,000 bases. However, this structural advantage introduces significant computational challenges, as long reads increase alignment complexity and memory consumption during overlap detection and error correction stages.

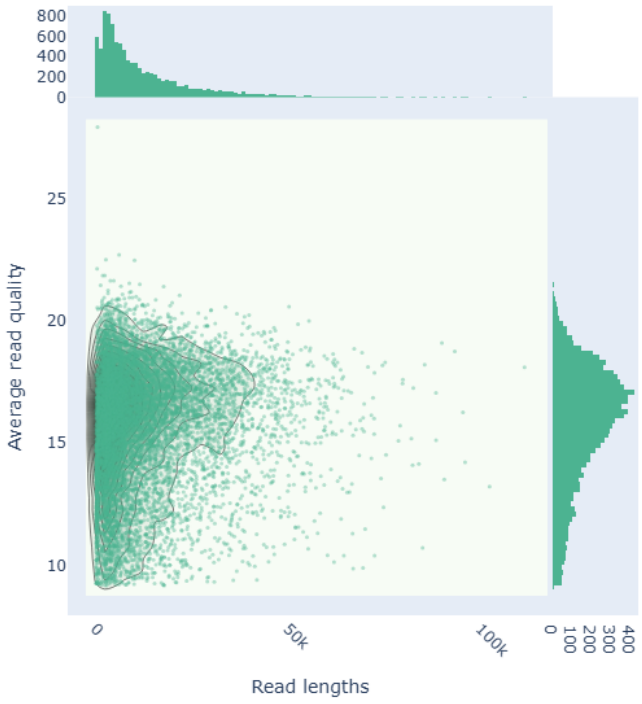

Figure 2. Quality and length distribution of ONT sequencing data

The scatter plot shows the relationship between read length and average read quality score, while the marginal histograms illustrate the distribution of read lengths and quality values. Most reads are concentrated in shorter length ranges with moderate quality scores. The observed variability in quality scores further confirms the noisy nature of Oxford Nanopore Technologies (ONT) data, which typically exhibits higher error rates compared with short-read sequencing technologies. Most reads were distributed within a quality score range of Q15–Q20 (Fig. 2, y-axis), corresponding to an estimated base accuracy of approximately 93–99%. As a result, computational pipelines must allocate substantial resources to error-correction algorithms before assembly graph construction can proceed. This characteristic directly influences HPC resource utilization, as the correction stage is predominantly CPU-intensive due to large-scale pairwise alignments, whereas downstream assembly phases become increasingly memory-dependent for graph construction and sorting operations. Therefore, the QC results not only validate the suitability of the dataset for genome assembly but also provide a clear explanation for the mixed CPU–memory workload profile observed during benchmarking.

3.2. Runtime Performance Across CPU–Memory Configurations

The execution time obtained from the nine benchmarking scenarios demonstrates clear performance differences associated with resource allocation strategies. Across all experiments, runtime ranged from approximately 51 hours to more than 109 hours (Fig. 3), indicating that computational performance was strongly influenced by the number of CPU cores assigned through the Slurm scheduler. Configurations using higher core counts consistently produced shorter execution times, while lower core allocations resulted in substantially longer runtimes regardless of memory settings (Table 3). This pattern suggests that the dominant computational burden during the NextDenovo workflow lies in highly parallelizable tasks, particularly during the read correction stage.

Figure 3. Runtime distribution across CPU core and memory configurations for NextDenovo assembly. Boxplots show that increasing CPU cores consistently reduces runtime, while memory allocation has a smaller effect, supporting the ANOVA result that CPU scaling is the primary driver of performance.

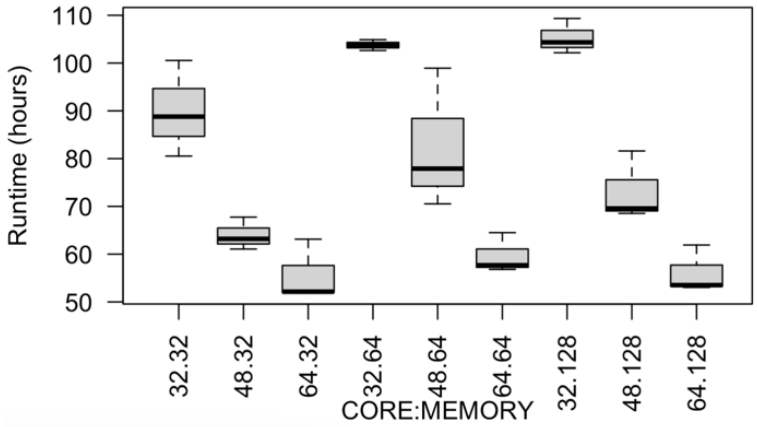

Figure 3. Empirical Runtime Distribution Across NextDenovo Resource Configurations

Table 3. Benchmarking Results Across CPU and Memory Configurations

Scenario	Core	RAM	Average Execution Time (HH:MM:SS)	Variance of Runtime (HH:MM:SS)	Average N50	Average Total Bases
1	64	128	56:09:20	1:02:23	6.866.154	868.275.373
2	64	64	59:39:40	0:44:21	7.776.054	864.557.353
3	64	32	55:42:40	1:43:20	7.362.535	864.341.400
4	48	128	73:13:40	2:12:04	7.349.549	868.046.778
5	48	64	82:27:20	9:03:01	7.527.335	865.337.056
6	48	32	64:00:20	0:29:08	7.303.823	862.213.174
7	32	128	105:17:03	0:33:46	7.360.544	868.107.570
8	32	64	103:45:40	0:03:13	7.440.986	866.198.503
9	32	32	89:57:20	4:13:00	8.027.225	863.782.011

When comparing memory configurations within the same CPU level, the influence of RAM allocation was observable but less pronounced. Increasing memory capacity occasionally reduced runtime, especially in scenarios where sorting and graph construction phases were active; however, the improvement was not consistently proportional to memory size. For example, configurations with 64 cores showed relatively similar runtimes between 64 GB and 128 GB memory, indicating that additional memory beyond a certain threshold did not produce significant acceleration. This observation suggests that memory over-provisioning may lead to inefficient resource utilization without providing substantial performance gains.

The variability observed among repeated runs under identical configurations further highlights the stochastic nature of long-read assembly workflows. Differences in runtime likely originate from internal algorithmic behaviors, I/O fluctuations, and scheduler-level resource contention, all of which are common in shared HPC environments. Nevertheless, the overall trends remained consistent across repetitions, supporting the robustness of the benchmarking results.

3.3. Statistical Analysis of Runtime Performance

To quantitatively evaluate the influence of computational resources on execution time, a two-way ANOVA was performed using CPU core count (CORE) and memory allocation (MEMORY) as independent variables and total runtime as the response variable. The analysis revealed that CPU core allocation had a highly significant effect on runtime ($F = 77.41$, $p < 0.001$), confirming that processing performance is primarily governed by parallel computational capacity. This finding aligns with the architecture of NextDenovo, where large-scale sequence alignment and correction tasks heavily rely on multi-threaded execution.

Memory allocation also exhibited a statistically significant effect on runtime ($F = 6.42$, $p = 0.007$), although its contribution was considerably smaller compared with CPU cores. This indicates that while sufficient memory is necessary to support graph construction and sorting operations, increasing memory beyond operational requirements provides diminishing returns in overall performance. The relatively smaller sum-of-squares contribution of memory further reinforces its secondary role in determining runtime efficiency.

Interestingly, the interaction between CPU cores and memory allocation was not statistically significant ($F = 1.31$, $p = 0.305$). This result suggests that the performance gain obtained from increasing CPU cores remains relatively stable across different memory configurations. In practical terms, CPU scaling behavior is largely independent of memory allocation within the tested range. From an HPC resource management perspective, this is an important finding because it indicates that users can optimize CPU allocation

without requiring proportional increases in memory, thereby reducing the risk of resource over-provisioning.

To provide a comprehensive evaluation of the resource allocation trade-offs, a decision rationale matrix comparing the high-end processing configurations (64 CPU cores) across multi-dimensional performance vectors is formulated in Table 4.

Table 4. Rationale Matrix for Optimal Configuration Selection

Resource Metrics	Scenario 1 (Over-provisioned)	Scenario 2 (Optimal Balanced)	Scenario 3 (Under-provisioned RAM)
Allocated Cores / RAM	64 Cores / 128 GB	64 Cores / 64 GB	64 Cores / 32 GB
Avg Runtime	56:09:20	59:39:40	55:42:40
Runtime Variance	1:02:23	0:44:21 (Lowest)	1:43:20 (High)
Structural N50 Continuity	6.866.154	7.776.054 (Superior)	7.362.535
Stranded Cluster Capacity	Wasted (64 GB)	Zero (Fully Utilized)	Zero
Biological Validation Status	Not Tested	Validated (99.8% Score)	Not Tested

3.4. Assembly Quality Metrics Under Different Resource Settings

Beyond computational efficiency, genome assembly quality was evaluated to ensure that performance optimization did not compromise biological accuracy. Structural continuity was assessed using the N50 metric, while total assembled bases were used to examine genome completeness (Table 3). Across all benchmarking scenarios, N50 values remained relatively stable, indicating that variations in CPU and memory allocation did not significantly alter assembly continuity. Similarly, total assembled genome sizes showed minimal fluctuation, suggesting consistent reconstruction outcomes regardless of resource configuration.

These results demonstrate that reducing computational resources does not necessarily degrade assembly quality, provided that minimum operational thresholds are maintained. In particular, moderate memory configurations produced assembly metrics comparable to higher-memory runs, reinforcing the observation that excessive memory allocation

may not yield measurable biological benefits. This finding supports the hypothesis that careful tuning of HPC resources can maintain assembly quality while improving overall system efficiency.

3.5. Biological Validation and Optimal Configuration Selection

Following computational benchmarking, biological validation was performed on the selected optimal configuration to ensure that resource optimization did not compromise genome completeness. Based on runtime performance, assembly continuity metrics, and resource efficiency, Scenario 2 (64 CPU cores and 64 GB memory) was identified as the most balanced configuration. This scenario achieved near-minimum execution time while avoiding excessive memory allocation compared with higher-resource settings.

The assembly produced under Scenario 2 was evaluated using BUSCO against the embryophyta_odb10 lineage dataset containing 1,614 conserved plant genes. BUSCO provides a standardized measure of biological completeness by classifying orthologs into several categories: Complete (C), which represents genes fully recovered in the assembly; Complete and Single-Copy (S), indicating orthologs found once as expected; Complete and Duplicated (D), representing genes detected more than once; Fragmented (F), indicating partially recovered genes; and Missing (M), corresponding to orthologs not detected in the assembly.

The analysis of the Scenario 2 draft genome yielded a completeness score of 99.8%, successfully reconstructing 1,610 out of 1,614 essential genes (Table 4). This result is high. Furthermore, the analysis detected a 10.2% gene duplication rate (165 genes). In the context of the Fabaceae family, this duplication is not a computational error but a reflection of the evolutionary history of the Acacia genus, which is characterized by ancestral Whole Genome Duplication events [21], [22]. These biological findings confirm that the 64 GB RAM configuration was sufficient to preserve the complex genetic architecture of *A. crassicarpa*. The negligible amount of missing (0.2%) and fragmented (0.1%) genes proves that the computational decision to avoid over-provisioning did not result in the loss of critical genetic information. Consequently, the assembly produced under Scenario 2 is not only computationally efficient but also biologically robust and

suitable for downstream applications such as phylogenetics and structural gene annotation.

Table 5. BUSCO assessment results for the genome assembly generated in Scenario 2

BUSCO Category	Description	Count	Percentage
Complete (C)	Fully recovered conserved genes	1,610	99.8
Complete Single-Copy (S)	Complete genes present once	1,445	89.5
Complete Duplicated (D)	Complete genes present multiple times	165	10.2
Fragmented (F)	Partially recovered genes	1	0.1
Missing (M)	Conserved genes not detected	3	0.2
Total	Empryophyta lineage dataset	1,614	100

Although BUSCO analysis was performed only on the selected optimal scenario, the consistency observed in structural assembly metrics (N50 and total assembled bases) across benchmarking experiments suggests that quality remained stable under different computational configurations. Therefore, the BUSCO validation supports the conclusion that efficient HPC resource allocation can be achieved without compromising biological integrity.

3.6. Synthesis of Resource-Performance Trade-offs and Analytical Discussion

The experimental findings reveal a critical threshold in computational resource allocation where further scaling no longer yields proportional benefits for genome assembly. The high significance of CPU cores ($p < 0.001$) in reducing runtime proves that the NextDenovo workflow is primarily CPU-bound, specifically during the highly parallelizable read-correction phase. Conversely, the diminishing returns observed when increasing memory beyond 64 GB suggest that the assembly of the 900 Mb *A. crassicarpa* genome reaches a memory saturation point, where additional RAM becomes "stranded capacity" that increases system waiting time without enhancing N50 continuity or biological completeness.

The performance tradeoffs detailed in Table 4 demonstrate that, although Scenario 3 offered a slightly shorter execution time (55:42:40), it suffered from a high runtime variance (1:43:20). This severe instability is caused by memory thrashing and disk swapping when running complex graph calculations near the absolute minimum boundary of 32 GB RAM in a shared cluster architecture. On the other hand, allocating 128 GB in Scenario 1 resulted in stranded computational blocks without acceleration and led to lower continuity (N50 of 6.86 Mb) . Therefore, Scenario 2 (64 cores, 64 GB RAM) was selected as the optimal configuration because it provided a more robust balance between runtime efficiency, minimized variance, and superior structural stability. The biological validation through BUSCO, which achieved a 99.8% completeness score, empirically confirms that mitigating over-provisioning does not compromise genetic integrity. This synthesis underscores that benchmarking is essential to avoid inefficient resource monopolization in shared HPC environments, ensuring that computational budgets are allocated based on empirical performance limits rather than over-estimated safety margins.

4. CONCLUSION

This study concludes that computational resource allocation significantly influences the efficiency of long-read genome assembly, with CPU scalability acting as the primary driver for runtime reduction. Through systematic benchmarking across nine configurations, the results demonstrate that the configuration of 64 CPU cores and 64 GB RAM (Scenario 2) provides the optimal balance, achieving an efficient execution time of 59 hours 39 minutes 40 seconds and a high biological completeness score of 99.8%. These findings highlight that increasing resources beyond operational requirements particularly memory beyond 64 GB does not improve assembly quality and leads to inefficient resource over-provisioning. This research provides a practical reference for optimizing HPC throughput without compromising biological integrity, as evidenced by stable N50 and BUSCO metrics. However, a primary limitation of this study is that complete biological validation via BUSCO was strictly conducted for the selected optimal scenario, while other resource allocation treatments were contrasted primarily using computational runtime and structural assembly metrics. Furthermore, it is important to acknowledge that these empirical indicators are presently limited to the NextDenovo assembler, the A. crassicarpa dataset, and the specific architecture of the Mahameru HPC

environment. Consequently, future work must proceed with a cautious approach by systematically evaluating additional long-read assemblers (such as Flye or Canu), incorporating post-assembly polishing stages, testing varied genomic datasets, and benchmarkings across diverse alternative HPC hardware configurations to further support the automated resource prediction strategies and reinforce the sustainable utilization of national shared HPC infrastructures.

ACKNOWLEDGMENT

The authors would like to acknowledge the Mahameru HPC facility provided by the National Research and Innovation Agency, Indonesia, for the computational resources and technical support that enabled this research. The availability of this national-scale infrastructure was essential for conducting large-scale genome assembly benchmarking and analysis.

REFERENCES

[1] A. E. Ahmed *et al.*, "Design considerations for workflow management systems use in production genomics research and the clinic," *Sci. Rep.*, vol. 11, no. 1, 2021, doi: 10.1038/s41598-021-99288-8.

[2] J. Mariette *et al.*, "Jflow: A workflow management system for web applications," *Bioinformatics*, vol. 32, no. 3, pp. 456–458, 2016, doi: 10.1093/bioinformatics/btv589.

[3] N. Mujkanovic, J. J. Durillo, N. Hammer, and T. Müller, "Survey of adaptive containerization architectures for HPC," in *ACM Int. Conf. Proc. Ser.*, Association for Computing Machinery, 2023, pp. 165–176. doi: 10.1145/3624062.3624588.

[4] J. Rybicki and C. Böttcher, "Data Logistics Service in eFlows4HPC," in *ICT Electron. Conv., MIPRO - Proc.*, Babic S., Car Z., Cicin-Sain M., Cisic D., Ergovic P., Grbac T.G., Gradisnik V., Gros S., Jokic A., Jovic A., Jurekovic D., Katulic T., Koricic M., Mornar V., Petrovic J., Skala K., Skvorc D., Sruk V., Svaco M., Tijan E., Vrcek N., and Vrdoljak B., Eds., Institute of Electrical and Electronics Engineers Inc., 2024, pp. 892–897. doi: 10.1109/MIPRO60963.2024.10569664.

[5] M. Jiang, C. Bu, J. Zeng, Z. Du, and J. Xiao, "Applications and challenges of high performance computing in genomics," *CCF Trans. High Perform. Comput.*, vol. 3, no. 4, pp. 344–352, 2021, doi: 10.1007/s42514-021-00081-w.

[6] J. I. Diaz-Riaño and J. Duitama, "Current Progress in Phased Genome Assembly from Long-Read DNA Sequencing Data," in *Methods Mol. Biol.*, vol. 2955, Humana Press Inc., 2025, pp. 51–70. doi: 10.1007/978-1-0716-4702-8_4.

[7] P. Morisse, T. Lecroq, and A. Lefebvre, "Hybrid correction of highly noisy long reads using a variable-order de Bruijn graph," *Bioinformatics*, vol. 34, no. 24, pp. 4213–4222, 2018, doi: 10.1093/bioinformatics/bty521.

[8] O. Bhowmik, T. Rahman, and A. Kalyanaraman, "Maptcha: an efficient parallel workflow for hybrid genome scaffolding," *BMC Bioinformatics*, vol. 25, no. 1, 2024, doi: 10.1186/s12859-024-05878-4.

[9] L. Obinu, T. Booth, H. De Weerd, U. Trivedi, and A. Porceddu, "Colora: a Snakemake workflow for complete chromosome-scale de novo genome assembly," *Bioinformatics*, vol. 41, no. 5, 2025, doi: 10.1093/bioinformatics/btaf175.

[10] V. Sanz, A. Pousa, M. Naiouf, and A. De Giusti, "A Fast and Scalable Genomic Data Compressor for Multicore Clusters," in *Lect. Notes Comput. Sci.*, Lees M.H., Cai W., Cheong S.A., Su Y., Abramson D., Dongarra J.J., and Sloot P.M.A., Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 180–188. doi: 10.1007/978-3-031-97635-3_22.

[11] S. S. Sunkara, E. Abeysinghe, C. Langin, S. Pamidighantam, M. Pierce, and S. Marru, "Simplifying access to campus resources at Southern Illinois University with a science gateway," in *ACM Int. Conf. Proc. Ser.*, Association for Computing Machinery, 2018. doi: 10.1145/3219104.3229252.

[12] A. Tyryshkina, N. Coraor, and A. Nekrutenko, "Predicting runtimes of bioinformatics tools based on historical data: Five years of Galaxy usage," *Bioinformatics*, vol. 35, no. 18, pp. 3453–3460, 2019, doi: 10.1093/bioinformatics/btz054.

[13] J. Bader, F. Lehmann, L. Thamsen, U. Leser, and O. Kao, "Lotaru: Locally predicting workflow task runtimes for resource management on heterogeneous infrastructures," in *Future Gener Comput Syst*, Elsevier B.V., 2024, pp. 171–185. doi: 10.1016/j.future.2023.08.022.

[14] J. Bader, F. Lehmann, L. Thamsen, J. Will, U. Leser, and O. Kao, "Lotaru: Locally Estimating Runtimes of Scientific Workflow Tasks in Heterogeneous Clusters," in

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

ACM Int. Conf. Proc. Ser., Pourabbas E., Zhou Y., Li Y., and Yang B., Eds., Association for Computing Machinery, 2022. doi: 10.1145/3538712.3538739.

[15] J. He and X. Liu, "Hybrid Teaching-Learning-Based Optimization for Workflow Scheduling in Cloud Environment," *IEEE Access*, vol. 11, pp. 100755–100768, 2023, doi: 10.1109/ACCESS.2023.3314735.

[16] M. Arif, A. Maurya, M. M. Rafique, D. S. Nikolopoulos, and A. R. Butt, "Application-Attuned Memory Management for Containerized HPC Workflows," in *Proc. - IEEE Int. Parallel Distrib. Process. Symp., IPDPS*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 114–127. doi: 10.1109/IPDPS57955.2024.00019.

[17] S. Bergman, O. Mutlu, W. Yong, K. Huang, and J. Zhang, "Composable Storage Servers: A Storage Paradigm for Disaggregated Systems," in *Int. Conf. Netw., Archit. Storage, NAS*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/NAS63802.2024.10781363.

[18] Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au, "Nanopore sequencing technology, bioinformatics and applications," *Nat. Biotechnol.*, vol. 39, no. 11, pp. 1348–1365, Nov. 2021, doi: 10.1038/s41587-021-01108-x.

[19] J. Hu *et al.*, "An efficient error correction and accurate assembly tool for noisy long reads," Bioinformatics, preprint, Mar. 2023. doi: 10.1101/2023.03.09.531669.

[20] H. Cheng, M. Asri, J. Lucas, S. Koren, and H. Li, "Scalable telomere-to-telomere assembly for diploid and polyploid genomes with double graph," *Nat. Methods*, vol. 21, no. 6, pp. 967–970, Jun. 2024, doi: 10.1038/s41592-024-02269-8.

[21] I. Massaro, R. S. Poethig, N. R. Sinha, and A. R. Leichty, "Chromosome-level genome of the transformable northern wattle, *Acacia crassicarpa*," *G3 Genes Genomes Genet.*, vol. 14, no. 3, p. jkad284, Mar. 2024, doi: 10.1093/g3journal/jkad284.

[22] X. Yue, Y. Yu, W. Gao, S. Chen, Z. Weng, and G. Ye, "Complete chloroplast genome sequence of *Acacia crassicarpa* (Fabaceae)," *Mitochondrial DNA Part B*, vol. 6, no. 8, pp. 2249–2250, Aug. 2021, doi: 10.1080/23802359.2021.1944365.

[23] J. Hu *et al.*, "NextDenovo: an efficient error correction and accurate assembly tool for noisy long reads," *Genome Biol.*, vol. 25, no. 1, p. 107, Apr. 2024, doi: 10.1186/s13059-024-03252-4.

[24] R. R. Wick and K. E. Holt, "Benchmarking of long-read assemblers for prokaryote whole genome sequencing." Feb. 01, 2021.

[25] T. Wirahman *et al.*, "Performance Evaluation of NAS Parallel and High-Performance Conjugate Gradient Benchmarks in Mahameru," *J. Online Inform.*, vol. 10, no. 2, pp. 248–259, Aug. 2025, doi: 10.15575/join.v10i2.1557.

[26] H. Li and R. Durbin, "Genome assembly in the telomere-to-telomere era," *Nat. Rev. Genet.*, vol. 25, no. 9, pp. 658–670, Sep. 2024, doi: 10.1038/s41576-024-00718-w.

[27] W. De Coster and R. Rademakers, "NanoPack2: population-scale evaluation of long-read sequencing data," *Bioinformatics*, vol. 39, no. 5, p. btad311, May 2023, doi: 10.1093/bioinformatics/btad311.

[28] A. A. Jauhal and R. D. Newcomb, "Assessing genome assembly quality prior to downstream analysis: N50 versus BUSCO," *Mol. Ecol. Resour.*, vol. 21, no. 5, pp. 1416–1421, Jul. 2021, doi: 10.1111/1755-0998.13364.

[29] M. Manni, M. R. Berkeley, M. Seppey, and E. M. Zdobnov, "BUSCO: Assessing Genomic Data Quality and Beyond," *Curr. Protoc.*, vol. 1, no. 12, Dec. 2021, doi: 10.1002/cpz1.323.