JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Implementasi dan Analisis Kinerja Komputasi Paralel Menggunakan Ray pada Lingkungan Multi-Core untuk Pemrosesan Audio

Phie Chyan*, Sean Coonery Sumarta

Fakultas Teknologi Informasi, Program Studi Informatika, Universitas Atma Jaya Makassar, Makassar, Indonesia
Email: [1*]phie_chyan@lecturer.uajm.ac.id , [2]Sean_sumarta@lecturer.uajm.ac.id
Email Penulis Korespondensi: phie_chyan@lecturer.uajm.ac.id
Submitted **07-03-2026;** Accepted **02-06-2026;** Published **30-06-2026**

Abstrak

Pemanfaatan komputasi paralel menjadi salah satu pendekatan yang efektif untuk meningkatkan efisiensi pemrosesan data, khususnya pada aplikasi yang bersifat CPU-bound. Penelitian ini bertujuan untuk mengimplementasikan komputasi paralel menggunakan framework Ray pada proses ekstraksi fitur audio dalam lingkungan multi-core berbasis single-node. Dataset yang digunakan terdiri dari 1000 file audio berekstensi .wav dari RAVDESS yang diproses melalui ekstraksi fitur pada domain waktu, frekuensi, dan waktu–frekuensi menggunakan pustaka Librosa. Fitur yang diekstraksi meliputi zero-crossing rate, spectral features, dan Mel-Frequency Cepstral Coefficients (MFCC) yang merepresentasikan karakteristik sinyal audio secara komprehensif. Setiap file audio diperlakukan sebagai satu unit pekerjaan (task) yang didistribusikan ke beberapa worker untuk diproses secara paralel. Pendekatan ini memungkinkan pemrosesan data dilakukan secara independen tanpa ketergantungan antar task. Pengujian dilakukan dengan variasi jumlah worker sebanyak 1 hingga 4 untuk mengamati pengaruhnya terhadap waktu eksekusi. Parameter utama yang diamati adalah waktu eksekusi total dalam menyelesaikan seluruh proses ekstraksi fitur. Hasil penelitian menunjukkan bahwa penerapan komputasi paralel mampu menurunkan waktu eksekusi dibandingkan dengan proses serial. Namun demikian, peningkatan kinerja yang dihasilkan tidak selalu bersifat linear akibat adanya overhead sistem dan keterbatasan sumber daya perangkat keras yang digunakan. Temuan ini menunjukkan bahwa pendekatan task parallelism menggunakan Ray dapat menjadi solusi praktis dalam meningkatkan efisiensi pemrosesan data audio pada lingkungan dengan sumber daya terbatas tanpa memerlukan infrastruktur komputasi terdistribusi yang kompleks.

Kata Kunci: Komputasi Paralel; Ekstraksi Fitur Audio; Task Parallelism; Pemrosesan Sinyal Audio; Lingkungan Multi-core

Abstract

Parallel computing has become an effective approach to improving data processing efficiency, particularly for CPU-bound applications. This study aims to implement parallel computing using the Ray framework for audio feature extraction in a multi-core single-node environment. The dataset consists of 1000 .wav audio files from the RAVDESS dataset, which are processed through feature extraction in the time, frequency, and time–frequency domains using the Librosa library. The extracted features include zero-crossing rate, spectral features, and Mel-Frequency Cepstral Coefficients (MFCC), which provide a comprehensive representation of audio signal characteristics. Each audio file is treated as an independent task and distributed across multiple workers for parallel processing. This approach allows data processing to be performed independently without inter-task dependencies. Experiments are conducted by varying the number of workers from 1 to 4 to observe their impact on execution time. The primary parameter observed is the total execution time required to complete the feature extraction process. The results show that the implementation of parallel computing reduces execution time compared to serial processing. However, the performance improvement is not strictly linear due to system overhead and hardware resource limitations. These findings indicate that a task parallelism approach using Ray can serve as a practical solution to improve audio data processing efficiency in resource-constrained environments without requiring complex distributed computing infrastructure.

Keywords: Parallel Computation; Audio Feature Extraction; Task Parallelism; Audio Signal Processing; Multi-core Environment

1. PENDAHULUAN

Pemanfaatan sumber daya komputasi pada sistem *multi-core* menjadi salah satu pendekatan yang semakin penting dalam meningkatkan efisiensi pemrosesan data, khususnya pada aplikasi yang bersifat *CPU-intensive* [1], [2]. Pada banyak kasus, sistem dengan spesifikasi perangkat keras yang terbatas tetap dituntut untuk mampu menangani beban kerja komputasi yang relatif tinggi, sehingga diperlukan strategi pemanfaatan sumber daya yang lebih optimal tanpa bergantung pada infrastruktur komputasi berskala besar [3], [4].

Salah satu pendekatan yang umum digunakan adalah komputasi paralel, di mana proses komputasi dibagi ke dalam beberapa *task* yang dapat dieksekusi secara bersamaan. Pendekatan ini telah banyak diimplementasikan menggunakan berbagai metode seperti *multiprocessing* pada Python, *Message Passing Interface* (MPI), serta *OpenMP* [5], [6]. Meskipun metode-metode tersebut terbukti efektif dalam meningkatkan performa komputasi, implementasinya seringkali memerlukan pengelolaan yang lebih kompleks terkait pembagian task, sinkronisasi proses, serta komunikasi antar proses, terutama pada lingkungan dengan sumber daya terbatas [7], [8].

Beberapa penelitian sebelumnya telah menunjukkan bahwa penerapan komputasi paralel pada berbagai jenis *workload* dapat memberikan peningkatan kinerja yang signifikan, khususnya pada proses komputasi numerik dan pemrosesan data berskala besar [9]–[11]. Studi-studi tersebut umumnya berfokus pada lingkungan komputasi berkinerja tinggi (*High Performance Computing*) atau sistem terdistribusi, sementara implementasi pada lingkungan yang lebih sederhana seperti *single-node* dengan arsitektur *multi-core* masih relatif terbatas [12]. Hal ini menunjukkan adanya kebutuhan untuk mengkaji pendekatan yang lebih praktis dan mudah diimplementasikan pada lingkungan komputasi dengan sumber daya terbatas [13], [14].

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Dalam praktiknya, banyak pengembang menghadapi kesulitan dalam mengimplementasikan komputasi paralel secara efektif, khususnya ketika harus mengelola distribusi beban kerja dan koordinasi antar proses secara manual. Oleh karena itu, diperlukan pendekatan yang lebih praktis dan fleksibel yang dapat diimplementasikan tanpa memerlukan konfigurasi sistem yang kompleks.

Framework berbasis Python seperti Ray menyediakan abstraksi yang memungkinkan pengembang untuk mengimplementasikan komputasi paralel berbasis *task* dengan lebih sederhana [15]. Dengan dukungan mekanisme *scheduling* otomatis dan pengelolaan task yang terintegrasi, Ray menjadi salah satu alternatif yang menarik untuk digunakan dalam lingkungan komputasi terbatas. Namun demikian, pemanfaatan Ray dalam konteks implementasi praktis pada lingkungan *multi-core* masih belum banyak dikaji secara mendalam, terutama pada *workload* yang bersifat *CPU-bound*.

Salah satu contoh *workload* yang relevan adalah pemrosesan data *audio*, yang melibatkan berbagai operasi komputasi seperti transformasi sinyal dan ekstraksi karakteristik statistik [16]. Proses ini umumnya bersifat independen antar data, sehingga berpotensi untuk dioptimalkan melalui pendekatan paralel. Dalam konteks ini, penting untuk memahami bagaimana implementasi komputasi paralel dapat diterapkan secara efektif serta bagaimana pengaruhnya terhadap kinerja sistem secara keseluruhan.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi pemanfaatan komputasi paralel dalam pemrosesan data *audio* pada lingkungan *multi-core*. Penelitian ini secara khusus menitikberatkan pada aspek implementasi dan pemanfaatan sistem, bukan pada analisis skalabilitas secara mendalam, sehingga memberikan perspektif yang berbeda dibandingkan studi yang berfokus pada evaluasi performa komputasi secara teoritis**.** Fokus *penelitian* diarahkan pada bagaimana mekanisme paralelisasi diterapkan serta bagaimana variasi jumlah *worker* memengaruhi kinerja sistem dari sisi waktu eksekusi dan karakteristik pemrosesan.

Kontribusi utama dari penelitian ini adalah penyajian pendekatan implementasi komputasi paralel yang praktis pada lingkungan dengan sumber daya terbatas, serta evaluasi kinerja sistem dalam konteks pemrosesan data *audio*. Hasil penelitian ini diharapkan dapat menjadi referensi dalam pengembangan sistem komputasi paralel yang efisien dan mudah diimplementasikan pada berbagai skenario aplikasi tanpa memerlukan infrastruktur komputasi berskala besar.

2. METODOLOGI PENELITIAN

2.1 Desain Penelitian dan Lingkungan Sistem

Penelitian ini dirancang untuk mengimplementasikan dan mengevaluasi pemanfaatan komputasi paralel dalam pemrosesan data audio pada lingkungan berbasis *multi-core*. Pendekatan yang digunakan berfokus pada penerapan mekanisme paralelisasi dalam skenario komputasi praktis, dengan tujuan untuk mengoptimalkan pemanfaatan sumber daya CPU tanpa melibatkan infrastruktur komputasi terdistribusi berskala besar.

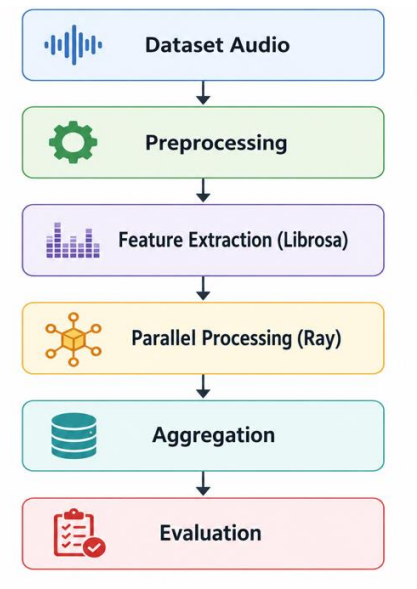
Tahapan penelitian dimulai dari persiapan dataset *audio*, dilanjutkan dengan proses ekstraksi fitur menggunakan pustaka Librosa, serta integrasi mekanisme komputasi paralel menggunakan *framework* Ray. Seluruh proses dirancang sebagai alur kerja yang terstruktur, di mana setiap tahap memiliki peran dalam mendukung eksekusi komputasi secara paralel. Diagram tahapan penelitian yang digunakan dalam studi ini ditunjukkan pada Gambar 1, yang menggambarkan alur proses mulai dari input data hingga pengumpulan hasil.

Eksperimen dilakukan pada lingkungan komputasi berbasis *virtual machine* yang dikonfigurasi untuk merepresentasikan sistem *multi-core* dalam satu node komputasi. Lingkungan ini dipilih karena mencerminkan kondisi yang umum digunakan dalam penelitian skala kecil hingga menengah, di mana ketersediaan *cluster multi-node* tidak selalu tersedia. Dengan pendekatan ini, penelitian dapat mengevaluasi efektivitas implementasi komputasi paralel dalam memanfaatkan sumber daya CPU yang terbatas.

Perangkat keras yang digunakan dalam penelitian ini berupa laptop dengan prosesor Intel Core i7-14650HX dan memori utama sebesar 16 GB RAM. *Virtual machine* dikonfigurasi untuk menggunakan 16 *virtual* CPU (vCPU) yang dialokasikan dari prosesor fisik, sehingga memungkinkan pemanfaatan sumber daya multi-core secara terkontrol dalam lingkungan eksperimen. Sistem operasi yang digunakan adalah Ubuntu, dengan lingkungan pemrograman Python yang telah dilengkapi dengan pustaka Ray dan Librosa sebagai komponen utama dalam implementasi sistem.

Seluruh eksperimen dijalankan dalam kondisi sistem yang terkontrol, di mana tidak terdapat proses komputasi berat lain yang berjalan secara bersamaan. Hal ini dilakukan untuk meminimalkan interferensi penggunaan CPU yang dapat memengaruhi hasil pengujian. Dengan konfigurasi lingkungan yang konsisten, hasil eksperimen diharapkan dapat direplikasi pada sistem dengan spesifikasi yang serupa.

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 1. Alur Tahapan Penelitian Komputasi Paralel pada Pemrosesan Data Audio

Tahapan penelitian yang ditunjukkan pada Gambar 1 terdiri dari beberapa langkah utama yang dilakukan secara berurutan. Tahap pertama adalah persiapan dataset *audio*, di mana data dikumpulkan dan disiapkan dalam format yang sesuai untuk proses pengolahan. Selanjutnya dilakukan tahap preprocessing untuk memastikan kualitas data yang digunakan dalam proses ekstraksi fitur.

Pada tahap berikutnya dilakukan proses ekstraksi fitur audio menggunakan pustaka Librosa. Proses ini menghasilkan representasi numerik dari sinyal audio yang akan digunakan dalam komputasi selanjutnya. Setelah itu, data yang telah diproses digunakan sebagai input dalam tahap komputasi paralel.

Tahap komputasi paralel dilakukan dengan mendistribusikan proses ekstraksi fitur ke beberapa worker menggunakan Ray. Setiap *worker* memproses data secara independen sesuai dengan pembagian tugas yang telah ditentukan. Hasil dari proses paralel ini kemudian dikumpulkan kembali pada tahap agregasi.

Tahap terakhir adalah evaluasi hasil komputasi, yang dilakukan dengan mengamati waktu eksekusi dan kinerja sistem secara keseluruhan. Seluruh tahapan ini dirancang untuk memastikan bahwa proses komputasi berjalan secara sistematis dan dapat direplikasi pada lingkungan yang serupa.

2.2 Arsitektur Sistem dan Implementasi Ray

Implementasi komputasi paralel dalam penelitian ini dilakukan dengan memanfaatkan *framework* Ray sebagai alat untuk mendistribusikan proses komputasi pada lingkungan *multi-core*. Fokus utama pada tahap ini adalah bagaimana proses ekstraksi fitur audio dapat dijalankan secara paralel melalui pembagian tugas ke beberapa *worker* yang tersedia.
Sistem yang dibangun terdiri dari satu program utama yang bertugas mengelola alur pemrosesan data. Program ini membaca *dataset audio*, kemudian membagi proses ekstraksi fitur menjadi sejumlah task yang dapat dijalankan secara bersamaan. Task tersebut selanjutnya didistribusikan ke *worker* yang tersedia untuk diproses secara independen.

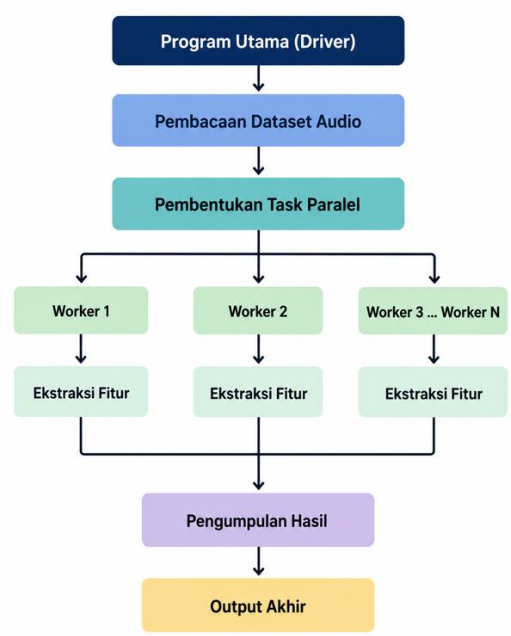
Dalam implementasinya, fungsi ekstraksi fitur *audio* didefinisikan sebagai fungsi yang dapat dieksekusi secara paralel menggunakan mekanisme yang disediakan oleh Ray. Setiap file audio diperlakukan sebagai satu unit pekerjaan, sehingga distribusi task dapat dilakukan secara otomatis tanpa memerlukan pengaturan manual yang kompleks.

Seluruh proses dijalankan dalam satu lingkungan komputasi, sehingga komunikasi antar proses berlangsung secara lokal tanpa melibatkan jaringan eksternal. Pendekatan ini dipilih untuk menyederhanakan implementasi sekaligus mengurangi kompleksitas sistem, terutama dalam pengujian pada lingkungan dengan sumber daya terbatas.

Secara konseptual, arsitektur sistem yang digunakan dalam penelitian ini ditunjukkan pada Gambar 2. Alur kerja sistem dimulai dari pembacaan data audio oleh program utama, dilanjutkan dengan pembentukan task paralel untuk setiap file. *Task* tersebut kemudian didistribusikan ke worker untuk diproses, dan hasilnya dikumpulkan kembali oleh program utama sebagai output akhir.

Dalam implementasinya, jumlah *worker* dikonfigurasi secara eksplisit sesuai dengan skenario eksperimen yang digunakan. Program utama menginisialisasi Ray berdasarkan jumlah CPU yang tersedia, kemudian mendistribusikan task sesuai dengan kapasitas tersebut. Setiap *worker* menjalankan proses ekstraksi fitur secara mandiri tanpa ketergantungan antar *task*, sehingga tidak memerlukan mekanisme sinkronisasi yang kompleks.

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 2. Arsitektur Sistem Komputasi Paralel untuk Pemrosesan Data Audio

2.3 Desain Beban Kerja

Desain beban kerja dalam penelitian ini difokuskan pada proses ekstraksi fitur audio sebagai representasi komputasi numerik intensif (*CPU-bound workload*). Beban kerja dirancang untuk mensimulasikan skenario pemrosesan sinyal yang umum digunakan dalam aplikasi pengenalan suara, klasifikasi emosi, dan analisis *audio* berbasis pembelajaran mesin.

Dataset yang digunakan terdiri dari 1000 file audio berekstensi *.wav* yang berasal dari dataset RAVDESS. Setiap file memiliki durasi antara 3 hingga 5 detik dengan format *uncompressed PCM*, sehingga tidak memerlukan proses dekompresi tambahan saat pemuatan data. Sampling rate mengikuti standar dataset, sehingga setiap file mengandung sejumlah besar sampel sinyal dalam domain waktu. Seluruh file diproses sebagai satu kesatuan beban kerja untuk menghasilkan volume komputasi yang cukup signifikan.

Proses ekstraksi fitur dilakukan menggunakan pustaka *Librosa* melalui pendekatan multi-domain yang mencakup domain waktu, domain frekuensi, dan domain waktu–frekuensi. Pada domain waktu, fitur yang dihitung meliputi *zero-crossing rate* sebagai representasi perubahan sinyal terhadap waktu. Secara sederhana, ZCR dapat dinyatakan sebagai:

$$ZCR = \frac{Nzc}{N} \qquad (1)$$

di mana *Nzc* adalah jumlah perubahan tanda antar sampel dan *N* adalah jumlah total sampel. Pada domain frekuensi, dilakukan transformasi menggunakan *Short-Time Fourier Transform (STFT)* untuk memperoleh fitur seperti *spectral centroid*, *spectral roll-off*, dan *spectral bandwidth*. Sementara itu, pada domain waktu–frekuensi digunakan *Mel-Frequency Cepstral Coefficients (MFCC)* yang diperoleh dari transformasi *Mel-spectrogram* [17], [18].

Seluruh fitur yang diperoleh diringkas menjadi representasi numerik melalui proses agregasi statistik terhadap frame sinyal. Proses ini melibatkan operasi komputasi seperti transformasi Fourier dan manipulasi matriks yang dilakukan secara berulang untuk setiap file audio, sehingga menghasilkan beban komputasi yang signifikan.

Berdasarkan karakteristik tersebut, beban kerja dalam penelitian ini dikategorikan sebagai *CPU-bound*, di mana sebagian besar waktu eksekusi digunakan untuk operasi komputasi numerik. Selain itu, setiap file audio dapat diproses secara independen tanpa ketergantungan antar task, sehingga termasuk dalam kategori *embarrassingly parallel workload*. Karakteristik ini menjadikan proses ekstraksi fitur audio sesuai untuk menguji implementasi komputasi paralel pada lingkungan *multi-core*.

2.4 Skenario dan Desain Eksperimen

Prosedur eksperimen dalam penelitian ini dirancang untuk menguji implementasi komputasi paralel pada proses ekstraksi fitur *audio* dalam lingkungan *multi-core.* Pengujian dilakukan dengan menjalankan sistem pada beberapa konfigurasi jumlah worker untuk mengamati pengaruhnya terhadap waktu eksekusi proses.

Eksperimen dilakukan secara bertahap dengan variasi jumlah worker sebanyak 1, 2, 3, dan 4. Setiap konfigurasi digunakan untuk menjalankan proses ekstraksi fitur terhadap seluruh dataset yang terdiri dari 1000 file *audio*. Pengujian dilakukan secara berulang untuk setiap konfigurasi guna memastikan konsistensi hasil yang diperoleh.

Pada setiap percobaan, sistem akan membaca seluruh file audio, kemudian mendistribusikan proses ekstraksi fitur ke *worker* sesuai dengan jumlah yang telah ditentukan. Setiap *worker* memproses sejumlah file secara independen hingga seluruh dataset selesai diproses. Setelah proses selesai, hasil komputasi dikumpulkan kembali oleh program utama.

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Pengukuran waktu eksekusi dilakukan dengan mencatat durasi total yang dibutuhkan sejak proses dimulai hingga seluruh file selesai diproses. Proses ini dilakukan menggunakan fungsi pengukuran waktu yang tersedia dalam Python untuk memperoleh nilai waktu yang akurat.

Untuk menjaga konsistensi hasil eksperimen, setiap pengujian dilakukan dalam kondisi sistem yang terkendali, di mana tidak terdapat proses komputasi lain yang berjalan secara bersamaan. Selain itu, konfigurasi sistem dan dataset yang digunakan dijaga tetap sama pada setiap percobaan. Melalui prosedur ini, penelitian dapat mengamati bagaimana implementasi komputasi paralel memengaruhi kinerja sistem dalam menyelesaikan beban kerja yang bersifat CPU-intensive, tanpa menitikberatkan pada analisis performa yang bersifat teoritis.

2.5 Pengukuran Kinerja Sistem

Evaluasi performa dalam penelitian ini dilakukan untuk mengukur efektivitas mekanisme paralelisasi Ray dalam menangani beban kerja ekstraksi fitur audio pada lingkungan virtualisasi *single-node*. Analisis difokuskan pada pengukuran waktu eksekusi total, perhitungan *speedup*, serta tingkat efisiensi pemanfaatan sumber daya CPU. Ketiga metrik ini dipilih karena umum digunakan dalam studi komputasi paralel untuk menilai karakteristik skalabilitas dan efektivitas sistem [19].

a. Waktu eksekusi total (Execution time) : Waktu eksekusi total didefinisikan sebagai durasi yang dibutuhkan sistem untuk memproses seluruh 1000 file audio dalam satu siklus eksperimen. Secara matematis, waktu eksekusi dinyatakan sebagai persamaan (2) di mana t_{start} adalah waktu saat proses ekstraksi dimulai dan t_{finish} adalah waktu ketika seluruh task selesai dieksekusi dan hasil telah dikembalikan ke driver. Nilai waktu eksekusi dihitung untuk dua kondisi utama, yaitu eksekusi serial (T_{serial}) dan eksekusi paralel dengan sejumlah worker tertentu ($T_{parallel}(P)$), di mana Pmenyatakan jumlah worker yang digunakan. Perbandingan antara kedua nilai ini menjadi dasar dalam analisis percepatan sistem.

$$T = t_{finish} - t_{start} \quad (2)$$

b. Speedup : *Speedup* digunakan untuk mengukur tingkat percepatan yang diperoleh dari penerapan komputasi paralel dibandingkan dengan eksekusi serial. Secara matematis, *speedup* didefinisikan pada persamaan (3) di mana $S(P)$adalah nilai *speedup* pada jumlah worker P. Nilai *speedup* ideal bersifat linear, yaitu $S(P) \approx P$, yang menunjukkan bahwa penambahan worker memberikan peningkatan performa yang proporsional. Namun, dalam praktiknya, nilai *speedup* dapat lebih kecil dari Pakibat adanya overhead penjadwalan task, sinkronisasi, serta kompetisi sumber daya CPU dalam lingkungan virtualisasi.

$$S(P) = \frac{T_{serial}}{T_{parallel(P)}} \quad (3)$$

c. Efisiensi dan analisis skalabilitas : Untuk mengevaluasi seberapa optimal pemanfaatan worker yang digunakan, dihitung metrik *efficiency* yang didefinisikan sebagai persamaan (4) di mana $E(P)$menyatakan tingkat efisiensi sistem pada jumlah worker P. Nilai efisiensi berada pada rentang 0 hingga 1, dengan nilai mendekati 1 menunjukkan bahwa sistem mampu memanfaatkan sumber daya paralel secara optimal. Penurunan nilai efisiensi seiring bertambahnya worker dapat mengindikasikan adanya *diminishing return* akibat overhead manajemen task dan pembagian CPU virtual.

$$E(P) = \frac{S(P)}{P} \quad (4)$$

Melalui analisis waktu eksekusi, *speedup*, dan *efficiency*, penelitian ini tidak hanya mengukur percepatan yang dicapai, tetapi juga mengevaluasi karakteristik skalabilitas Ray pada lingkungan virtualisasi *single-node*. Kombinasi ketiga metrik tersebut memberikan gambaran komprehensif mengenai efektivitas paralelisasi serta batas optimal pemanfaatan CPU dalam konteks beban kerja ekstraksi fitur audio yang bersifat CPU-intensive [20].

3. HASIL DAN PEMBAHASAN

3.1 Data Sampel dan Preprocessing Audio

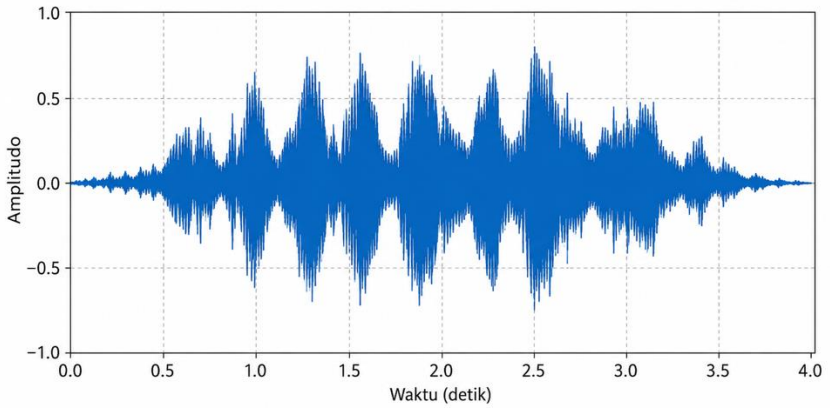
Dataset yang digunakan dalam penelitian ini terdiri dari file *audio* berekstensi *.wav* yang berasal dari dataset RAVDESS. Dataset ini dipilih karena menyediakan rekaman suara dengan kualitas yang baik serta banyak digunakan dalam penelitian pemrosesan sinyal audio dan analisis emosi suara. Setiap file audio memiliki durasi sekitar 3 hingga 5 detik dan direkam menggunakan format *uncompressed PCM*, sehingga data audio dapat diproses secara langsung tanpa memerlukan tahap dekompresi tambahan.

Pada tahap awal, file *audio* dibaca menggunakan pustaka *Librosa* untuk memperoleh representasi sinyal dalam domain waktu. Representasi ini berupa deret amplitudo sinyal terhadap waktu yang menggambarkan perubahan energi suara selama proses pengucapan berlangsung. Data sinyal tersebut kemudian digunakan sebagai input utama dalam proses ekstraksi fitur audio.

Sebagai contoh, salah satu sampel sinyal *audio* yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3. Gambar tersebut memperlihatkan bentuk gelombang (*waveform*) dari sinyal *audio* dalam domain waktu. Perubahan

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

amplitudo pada sinyal menunjukkan variasi karakteristik suara yang nantinya akan diproses lebih lanjut pada tahap ekstraksi fitur.



Gambar 3. Sinyal Audio dalam Domain Waktu

Berdasarkan Gambar 3, terlihat bahwa sinyal audio memiliki pola amplitudo yang berubah-ubah sepanjang durasi rekaman. Variasi ini merepresentasikan karakteristik akustik dari suara yang menjadi dasar dalam proses perhitungan fitur audio seperti *zero-crossing rate*, *spectral features*, dan *MFCC*. Sebelum proses ekstraksi fitur dilakukan, sinyal audio terlebih dahulu melewati tahap pemrosesan awal seperti pembacaan data dan normalisasi format agar dapat diproses secara konsisten oleh sistem.

Setelah tahap awal selesai dilakukan, setiap file audio diperlakukan sebagai satu unit pekerjaan yang akan diproses secara paralel menggunakan framework Ray. Pendekatan ini memungkinkan proses ekstraksi fitur dilakukan secara independen untuk setiap file audio, sehingga sesuai untuk diterapkan pada mekanisme *task parallelism* dalam lingkungan multi-core.

3.2 Implementasi Ekstraksi Fitur Audio Secara Paralel

Penerapan metode dalam penelitian ini dilakukan melalui proses ekstraksi fitur audio yang dijalankan secara paralel menggunakan framework *Ray*. Setiap file audio diperlakukan sebagai satu unit pekerjaan (*task*) yang diproses secara independen oleh *worker* yang tersedia. Pendekatan ini memungkinkan proses ekstraksi fitur dilakukan secara bersamaan pada beberapa file audio tanpa adanya ketergantungan antar proses.

Tahap awal dimulai dengan pembacaan file audio menggunakan pustaka *Librosa* untuk memperoleh representasi sinyal dalam domain waktu. Setelah sinyal berhasil dimuat, dilakukan proses ekstraksi fitur berdasarkan domain analisis yang telah ditentukan, yaitu domain waktu, domain frekuensi, dan domain waktu–frekuensi.

Salah satu fitur yang dihitung pada domain waktu adalah *zero-crossing rate (ZCR)*. Fitur ini digunakan untuk merepresentasikan jumlah perubahan tanda sinyal dalam interval waktu tertentu. Perhitungan *ZCR* dilakukan menggunakan Persamaan (1) pada bagian metodologi.
Sebagai ilustrasi sederhana, misalkan terdapat sinyal dengan 100 sampel dan ditemukan 10 perubahan tanda antar sampel berturut-turut. Maka nilai *ZCR* dapat dihitung sebagai:

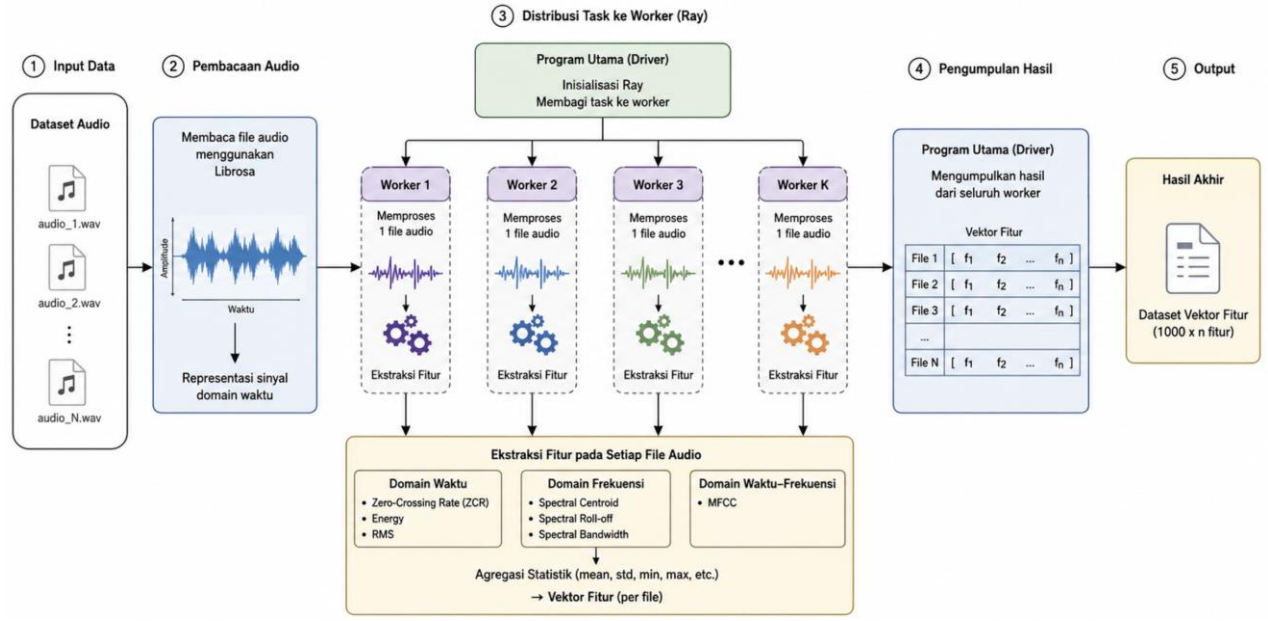
$$ZCR = \frac{10}{100} = 0.1$$

Nilai tersebut menunjukkan tingkat perubahan sinyal dalam domain waktu, yang dapat digunakan untuk merepresentasikan karakteristik tertentu dari audio. Selain fitur domain waktu, dilakukan pula ekstraksi fitur pada domain frekuensi menggunakan *Short-Time Fourier Transform (STFT)*. Transformasi ini menghasilkan representasi spektral yang digunakan untuk menghitung fitur seperti *spectral centroid*, *spectral roll-off*, dan *spectral bandwidth*. Sementara itu, pada domain waktu–frekuensi dilakukan perhitungan *Mel-Frequency Cepstral Coefficients (MFCC)* untuk memperoleh representasi karakteristik audio dalam skala mel.

Setelah seluruh fitur berhasil dihitung, hasil ekstraksi dari setiap file diringkas menjadi satu vektor fitur melalui proses agregasi statistik. Vektor fitur tersebut kemudian dikumpulkan kembali oleh program utama setelah seluruh *worker* menyelesaikan tugasnya masing-masing.

Alur penerapan metode yang digunakan dalam penelitian ini ditunjukkan pada Gambar 4. Gambar tersebut memperlihatkan tahapan mulai dari pembacaan sinyal *audio*, proses ekstraksi fitur, distribusi task ke *worker*, hingga pengumpulan hasil akhir.

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 4. Tahapan Penerapan Metode Ekstraksi Fitur Audio Secara Paralel

Berdasarkan Gambar 4, terlihat bahwa proses dimulai dari pembacaan data audio oleh program utama, kemudian dilanjutkan dengan pembentukan *task* paralel untuk setiap file. Task tersebut selanjutnya didistribusikan ke beberapa *worker* untuk menjalankan proses ekstraksi fitur secara independen. Setelah seluruh proses selesai, hasil dari masing-masing *worker* dikumpulkan kembali dan diringkas menjadi vektor fitur sebagai output akhir. Alur ini menunjukkan bahwa setiap tahapan berjalan secara sistematis dan sesuai dengan konsep *task parallelism* yang diterapkan dalam penelitian ini.

Dengan pendekatan ini, proses ekstraksi fitur dapat dijalankan secara paralel pada lingkungan *multi-core* tanpa memerlukan mekanisme sinkronisasi antar *worker*. Hal ini menunjukkan bahwa implementasi *task parallelism* menggunakan *Ray* dapat diterapkan secara praktis dalam pemrosesan data audio yang bersifat *CPU-bound*.

3.3 Hasil Pengujian

Hasil pengujian pada penelitian ini diperoleh dengan menjalankan proses ekstraksi fitur *audio* menggunakan variasi jumlah *worker* sebanyak 1, 2, 3, dan 4. Setiap konfigurasi digunakan untuk memproses seluruh dataset yang terdiri dari 1000 file audio, dengan tujuan untuk mengamati waktu eksekusi yang dihasilkan pada masing-masing kondisi.
Sebelum menampilkan hasil dalam bentuk visual, dilakukan pencatatan waktu eksekusi total untuk setiap konfigurasi *worker*. Hasil pengukuran tersebut disajikan dalam bentuk tabel untuk memberikan gambaran numerik terhadap kinerja sistem.

Tabel 1. Waktu eksekusi berdasarkan jumlah worker

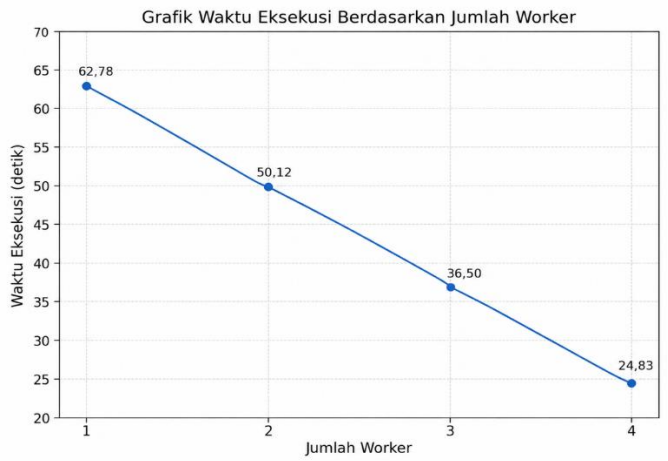
Jumlah *Worker*	Waktu Eksekusi (detik)
1	62,78
2	50,12
3	36,50
4	24,83

Berdasarkan Tabel 1, terlihat bahwa peningkatan jumlah *worker* memberikan penurunan waktu eksekusi dibandingkan dengan penggunaan satu *worker*. Hal ini menunjukkan bahwa pembagian beban kerja secara paralel mampu mempercepat proses komputasi dalam ekstraksi fitur audio.

Penurunan waktu eksekusi terjadi karena proses ekstraksi fitur pada setiap file audio dapat dijalankan secara independen tanpa ketergantungan antar task. Dengan karakteristik tersebut, distribusi beban kerja ke beberapa *worker* memungkinkan utilisasi CPU menjadi lebih optimal dibandingkan eksekusi secara serial.

Untuk memperjelas perbandingan hasil pengujian, data yang sama juga disajikan dalam bentuk grafik seperti yang ditunjukkan pada Gambar 5

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



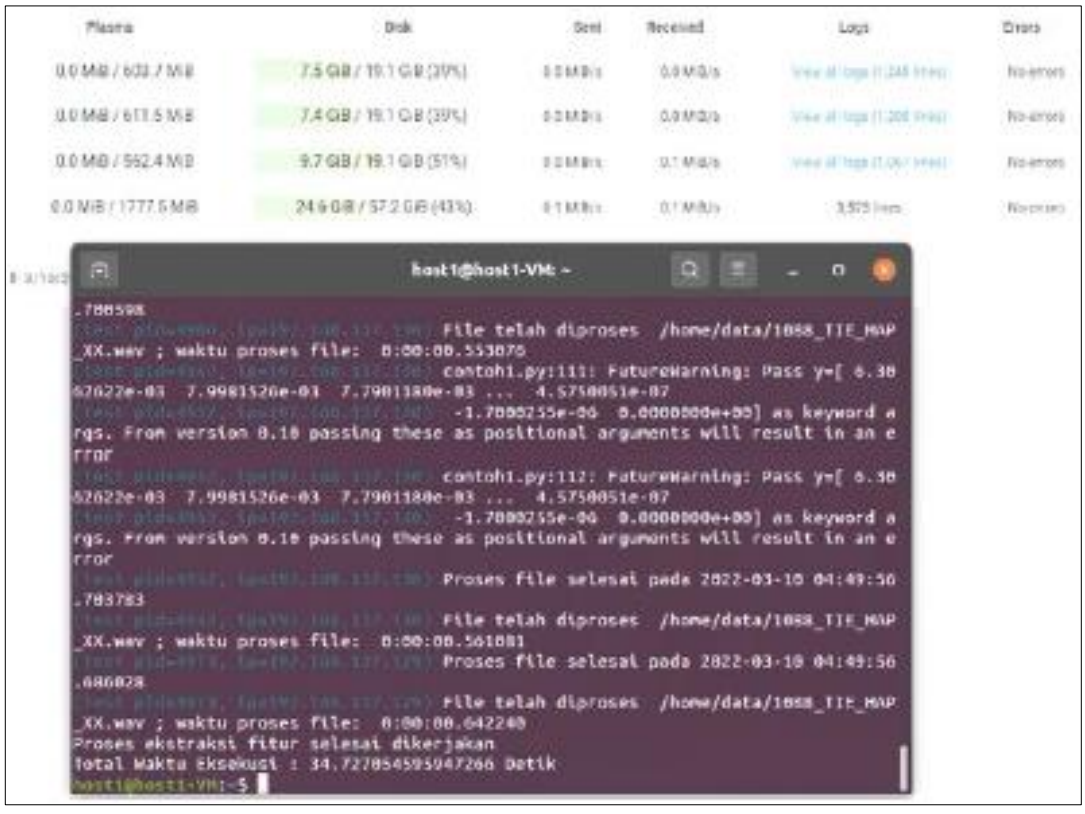
Gambar 5. Grafik Waktu Eksekusi terhadap Jumlah *Worker*

Grafik pada Gambar 5 menunjukkan kecenderungan penurunan waktu eksekusi seiring dengan bertambahnya jumlah *worker* yang digunakan. Namun demikian, penurunan tersebut tidak sepenuhnya bersifat linear. Pada konfigurasi tertentu, penambahan *worker* menghasilkan peningkatan kinerja yang lebih kecil dibandingkan konfigurasi sebelumnya.

Kondisi ini dapat dipengaruhi oleh beberapa faktor, seperti overhead dalam proses distribusi task, pengelolaan proses paralel oleh sistem, serta keterbatasan sumber daya CPU yang digunakan secara bersamaan. Selain itu, meskipun proses ekstraksi fitur bersifat *CPU-bound*, tetap terdapat waktu tambahan yang dibutuhkan sistem untuk melakukan inisialisasi *task* dan pengumpulan hasil dari setiap *worker*.

Hasil pengujian ini menunjukkan bahwa implementasi komputasi paralel menggunakan *Ray* pada lingkungan *multi-core* mampu meningkatkan kinerja pemrosesan data audio secara praktis. Temuan ini juga sejalan dengan beberapa penelitian sebelumnya yang menunjukkan bahwa pendekatan *task parallelism* efektif digunakan pada beban kerja yang bersifat independen antar task.

Dengan penyajian dalam bentuk tabel dan grafik, hasil pengujian dapat dianalisis secara lebih mudah, baik secara numerik maupun visual, sehingga memberikan gambaran yang lebih jelas mengenai kinerja sistem yang diimplementasikan. Selain hasil pengujian dalam bentuk tabel dan grafik, proses eksekusi sistem juga diamati secara langsung melalui *dashboard* Ray dan output terminal untuk memastikan bahwa mekanisme paralelisasi berjalan dengan baik. Salah satu contoh hasil eksekusi ditunjukkan pada Gambar 6.



Gambar 6. Dashboard Ray dengan Output Eksekusi pada Terminal

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 3, Juni 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i3.9649
Hal 899-908
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Dari Gambar 6 terlihat bahwa proses eksekusi sistem berhasil dijalankan melalui lingkungan Ray dan terminal. Output terminal menunjukkan proses ekstraksi fitur yang berjalan hingga menghasilkan waktu eksekusi total, sedangkan dashboard Ray memperlihatkan kondisi penggunaan sumber daya sistem selama proses berlangsung. Gambar ini mendukung hasil pengujian pada Tabel 1 dan Gambar 5 bahwa mekanisme paralelisasi telah berjalan sesuai dengan rancangan eksperimen.

3.4 Pembahasan

Berdasarkan hasil pengujian yang telah disajikan pada Tabel 1 dan Gambar 5, terlihat bahwa penerapan komputasi paralel memberikan pengaruh positif terhadap waktu pemrosesan ekstraksi fitur audio. Ketika proses dijalankan menggunakan satu *worker*, seluruh file audio diproses secara berurutan sehingga waktu eksekusi menjadi lebih tinggi. Setelah jumlah *worker* ditingkatkan, proses komputasi dapat dibagi ke beberapa unit eksekusi sehingga beberapa file audio dapat diproses secara bersamaan.

Penurunan waktu eksekusi ini menunjukkan bahwa beban kerja ekstraksi fitur audio memiliki karakteristik yang sesuai untuk diterapkan pada model *task parallelism*. Setiap file audio dapat diproses secara independen tanpa memerlukan pertukaran data antar file. Kondisi ini membuat proses pembagian tugas menjadi lebih sederhana karena setiap *worker* dapat menjalankan fungsi ekstraksi fitur secara mandiri. Dengan demikian, penggunaan *Ray* dalam penelitian ini mampu menyederhanakan implementasi komputasi paralel pada lingkungan multi-core.

Karakteristik pemrosesan audio dalam penelitian ini memungkinkan setiap file diproses secara independen tanpa memerlukan sinkronisasi antar proses. Kondisi tersebut membuat pendekatan *task parallelism* menjadi lebih mudah diterapkan dibandingkan model paralelisasi yang memerlukan pertukaran data secara terus-menerus antar *worker*. Oleh karena itu, pendekatan yang digunakan dalam penelitian ini relatif sesuai untuk lingkungan komputasi *multi-core* dengan sumber daya terbatas.

Meskipun penambahan jumlah *worker* mampu menurunkan waktu eksekusi, hasil pengujian juga menunjukkan bahwa peningkatan kinerja tidak selalu meningkat secara proporsional. Hal ini merupakan kondisi yang umum dalam komputasi paralel karena sistem tetap membutuhkan waktu tambahan untuk mengatur distribusi *task*, menjalankan proses paralel, serta mengumpulkan kembali hasil dari setiap *worker*. Dengan kata lain, semakin banyak *worker* yang digunakan, semakin besar pula kebutuhan sistem untuk mengelola proses eksekusi tersebut.

Faktor lain yang memengaruhi hasil pengujian adalah keterbatasan sumber daya pada lingkungan komputasi yang digunakan. Walaupun sistem dijalankan pada lingkungan *multi-core*, seluruh proses tetap berada dalam satu *node* komputasi. Oleh karena itu, setiap *worker* berbagi sumber daya yang sama, terutama CPU, memori, dan akses penyimpanan. Kondisi ini dapat menyebabkan peningkatan jumlah *worker* tidak selalu menghasilkan percepatan yang sebanding dengan jumlah *worker* yang ditambahkan.

Dari sisi implementasi, hasil penelitian ini menunjukkan bahwa *Ray* dapat digunakan sebagai solusi praktis untuk membagi proses komputasi ke beberapa *worker* tanpa memerlukan pengaturan paralelisasi yang terlalu kompleks. Hal ini menjadi penting terutama pada lingkungan penelitian atau pengembangan sistem dengan sumber daya terbatas, di mana penggunaan *cluster* besar atau infrastruktur komputasi terdistribusi belum tentu tersedia.

Hasil ini juga sejalan dengan prinsip umum komputasi paralel, yaitu bahwa beban kerja yang dapat dipisahkan menjadi beberapa tugas independen cenderung lebih mudah dipercepat melalui pembagian kerja. Pada konteks pemrosesan audio, proses ekstraksi fitur seperti *zero-crossing rate*, *spectral features*, dan *MFCC* dilakukan secara terpisah pada setiap file audio. Oleh karena itu, proses tersebut cocok untuk dijalankan secara paralel.

Namun demikian, hasil pengujian juga menunjukkan bahwa pemilihan jumlah *worker* tetap perlu disesuaikan dengan kondisi perangkat keras yang digunakan. Penggunaan jumlah *worker* yang terlalu sedikit belum mampu memanfaatkan sumber daya CPU secara optimal, sedangkan penggunaan jumlah *worker* yang terlalu banyak dapat meningkatkan beban pengelolaan proses. Dalam penelitian ini, konfigurasi 1 sampai 4 *worker* digunakan untuk menunjukkan bagaimana implementasi paralel memberikan peningkatan kinerja secara bertahap pada skala pengujian terbatas.

Penelitian ini masih terbatas pada lingkungan *single-node* dengan jumlah *worker* yang relatif kecil. Oleh karena itu, hasil yang diperoleh belum sepenuhnya menggambarkan perilaku sistem pada skala komputasi yang lebih besar atau pada lingkungan terdistribusi *multi-node.* Selain itu, pengujian difokuskan pada workload ekstraksi fitur audio sehingga karakteristik performa pada jenis workload lain masih perlu dikaji lebih lanjut.

Secara keseluruhan, hasil penelitian ini menunjukkan bahwa implementasi komputasi paralel menggunakan *Ray* dapat meningkatkan efisiensi pemrosesan data audio dibandingkan eksekusi secara serial. Temuan ini memperkuat bahwa pendekatan paralel berbasis *task* dapat menjadi alternatif praktis dalam pemrosesan data *audio* pada lingkungan *multi-core*, khususnya ketika beban kerja yang digunakan bersifat independen dan berulang.

4. KESIMPULAN

Penelitian ini berhasil mengimplementasikan komputasi paralel menggunakan framework *Ray* pada proses ekstraksi fitur *audio* dalam lingkungan multi-core berbasis *single-node*. Implementasi dilakukan dengan membagi proses ekstraksi fitur ke beberapa *worker* sehingga file *audio* dapat diproses secara paralel dan independen. Hasil pengujian menunjukkan bahwa penerapan komputasi paralel mampu menurunkan waktu eksekusi dibandingkan proses serial, yang menunjukkan

bahwa karakteristik workload ekstraksi fitur *audio* sesuai untuk diterapkan pada pendekatan *task parallelism*. Peningkatan jumlah *worker* memberikan peningkatan kinerja sistem, meskipun peningkatan tersebut tidak selalu bersifat linear akibat adanya *overhead* pengelolaan proses dan keterbatasan sumber daya perangkat keras yang digunakan secara bersamaan. Secara keseluruhan, hasil penelitian menunjukkan bahwa penggunaan *Ray* dapat menjadi solusi praktis untuk meningkatkan efisiensi pemrosesan data audio pada lingkungan dengan sumber daya terbatas tanpa memerlukan infrastruktur komputasi terdistribusi yang kompleks.

REFERENCES

[1] N. Perera *et al.*, "Supercharging distributed computing environments for high-performance data engineering," *Front. High Perform. Comput.*, vol. 2, p. 1384619, Jul. 2024, doi: 10.3389/FHPCP.2024.1384619.
[2] A. N. Hidayah, "Implementasi Parallel Computing untuk mengoptimalkan komputasi pada aplikasi transliterasi huruf latin ke Aksara Jawa," Universitas Islam Malang, 2018.
[3] P. Chyan and N. T. Saptadi, "Pemulihan Citra Berbasis Metode Markov Random Field," *JURIKOM (Jurnal Ris. Komputer)*, vol. 9, no. 2, p. 218, Apr. 2022, doi: 10.30865/JURIKOM.V9I2.3966.
[4] P. Chyan, "Design of intelligent camera-based security system with image enhancement support," *J. Phys. Conf. Ser.*, vol. 1341, no. 4, pp. 1–6, 2019, doi: 10.1088/1742-6596/1341/4/042009.
[5] G. Krawezik and F. Cappello, "Performance comparison of MPI and OpenMP on shared memory multiprocessors," *Concurr. Comput. Pract. Exp.*, vol. 18, no. 1, pp. 29–61, Jan. 2006, doi: 10.1002/CPE.905.
[6] X. He, K. Wang, Y. Feng, L. Lv, and T. Liu, "An implementation of MPI and hybrid OpenMP/MPI parallelization strategies for an implicit 3D DDG solver," *Comput. Fluids*, vol. 241, p. 105455, Jun. 2022, doi: 10.1016/J.COMPFLUID.2022.105455.
[7] P. Chyan *et al.*, *Pengantar Data Science: Mengambil Keputusan Berdasarkan Data*, vol. 1, no. 01. 2024. Accessed: May 15, 2026. [Online]. Available: https://jurnal.mifandimandiri.com/index.php/penerbitmmd/article/view/37
[8] K. Yoshida, S. Miwa, H. Yamaki, and H. Honda, "Analyzing the impact of CUDA versions on GPU applications," *Parallel Comput.*, vol. 120, p. 103081, Jun. 2024, doi: 10.1016/J.PARCO.2024.103081.
[9] B. Berisha, E. Mëziu, and I. Shabani, "Big data analytics in Cloud computing: an overview," *J. Cloud Comput.*, vol. 11, no. 1, 2022, doi: 10.1186/s13677-022-00301-w.
[10] R. W. Saputro, A. Aminuddin, and Y. Munarko, "Perbandingan Kinerja Komputasi Hadoop dan Spark untuk Memprediksi Cuaca (Studi Kasus : Storm Event Database)," *J. Repos.*, vol. 2, no. 4, pp. 463–474, Jan. 2020, doi: 10.22219/REPOSITOR.V2I4.30510.
[11] P. Samutrak*, J. Boonniyom, and J. Srisawat, "Parallel Matrix Multiplication On a Cluster of PCs," *Curr. Appl. Sci. Technol.*, vol. 5, no. 1, pp. 34–42, 2005, Accessed: May 15, 2026. [Online]. Available: https://li01.tci-thaijo.org/index.php/cast/article/view/160551
[12] S. Jin, Z. Huang, R. Diao, D. Wu, and Y. Chen, "Comparative Implementation of High Performance Computing for Power System Dynamic Simulations," *IEEE Trans. Smart Grid*, vol. 8, no. 3, pp. 1387–1395, May 2017, doi: 10.1109/TSG.2016.2647220.
[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. - Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/SPE.995.
[14] P. Chyan, R. Y. Carolus, and S. Mallu, "Emotion Recognition from Human Speech Using Linguistic and Paralinguistic Features Through a Soft Voting Approach," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, pp. 481–486, 2024, doi: 10.1109/EECSI63442.2024.10776080.
[15] P. Moritz *et al.*, *Ray: A Distributed Framework for Emerging {AI} Applications*. 2018. Accessed: May 15, 2026. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/nishihara
[16] M. S. Imran, A. F. Rahman, S. Tanvir, H. H. Kadir, J. Iqbal, and M. Mostakim, "An Analysis of Audio Classification Techniques using Deep Learning Architectures," *Proc. 6th Int. Conf. Inven. Comput. Technol. ICICT 2021*, pp. 805–812, Jan. 2021, doi: 10.1109/ICICT50816.2021.9358774.
[17] J. H. L. Hansen and S. Patil, "Speech under stress: Analysis, modeling and recognition," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4343 LNAI, pp. 108–137, 2007, doi: 10.1007/978-3-540-74200-5_6/SAVE-RESEARCH.
[18] R. Dillon and A. Ni Teoh, "Real-time Stress Detection Model and Voice Analysis: An Integrated VR-based Game for Training Public Speaking Skills," *IEEE Conf. Games*, pp. 1–4, 2021.
[19] E. Liang *et al.*, "RLlib: Abstractions for Distributed Reinforcement Learning," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 7, pp. 4768–4780, Dec. 2017, Accessed: May 15, 2026. [Online]. Available: https://arxiv.org/pdf/1712.09381
[20] A. Défossez, G. Synnaeve, and Y. Adi, "Real time speech enhancement in the waveform domain," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, 2020.