RESEARCH ARTICLE • OPEN ACCESS

Analisis Kinerja dan Efisiensi Energi *k-means* dan *Gaussian Mixture Model* Terdistribusi pada Klaster *Single Board Computer* dan *Personal Computer* dengan *Apache Spark*

Deffin Purnama Noer, Muhaza Liebenlito, dan Taufik Edy Sutanto

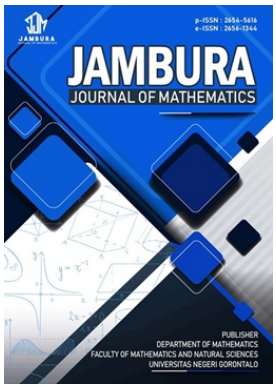
JAMBURA JOURNAL OF MATHEMATICS

Volume 8, Issue 1, Pages 32–44, February 2026

Diterima 8 Oktober 2025, Direvisi 10 Desember 2025, Disetujui 16 Desember 2025, Diterbitkan 7 Januari 2026

To Cite this Article : D. P. Noer, M. Liebenlito, dan T. E. Sutanto,"Analisis Kinerja dan Efisiensi Energi *k-means* dan *Gaussian Mixture Model* Terdistribusi pada Klaster *Single Board Computer* dan *Personal Computer* dengan *Apache Spark* ", *Jambura J. Math*, vol. 8, no. 1, pp. 32–44, 2026, *https://doi.org/10.37905/jjom.v8i1.33549*

JOURNAL INFO • JAMBURA JOURNAL OF MATHEMATICS



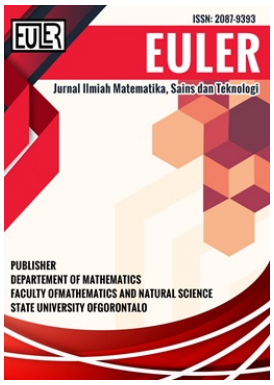
Homepage	:	http://ejurnal.ung.ac.id/index.php/jjom/index
Journal Abbreviation	:	Jambura J. Math.
Frequency	:	Biannual (February and August)
Publication Language	:	English (preferable), Indonesia
DOI	:	https://doi.org/10.37905/jjom
Online ISSN	:	2656-1344
Editor-in-Chief	:	Hasan S. Panigoro
Publisher	:	Department of Mathematics, Universitas Negeri Gorontalo
Country	:	Indonesia
OAI Address	:	http://ejurnal.ung.ac.id/index.php/jjom/oai
Google Scholar ID	:	iWLjgaUAAAAJ
Email	:	info.jjom@ung.ac.id

JAMBURA JOURNAL • FIND OUR OTHER JOURNALS

Jambura Journal of Biomathematics

Jambura Journal of Mathematics Education

Jambura Journal of Probability and Statistics



EULER : Jurnal Ilmiah Matematika, Sains, dan Teknologi

Jambura Journal of Mathematics, Volume 8, Issue 1, Pages 32–44, February 2026 https://doi.org/10.37905/jjom.v8i1.33549

Research Article

Check for updates

Analisis Kinerja dan Efisiensi Energi *k-means* dan *Gaussian Mixture Model* Terdistribusi pada Klaster *Single Board Computer* dan *Personal Computer* dengan *Apache Spark*

Deffin Purnama Noer[1], **Muhaza Liebenlito**[1,*], **Taufik Edy Sutanto**[1]

[1]*Program Studi Matematika, Universitas Islam Negeri Syarif Hidayatullah Jakarta, Tangerang Selatan 15412, Indonesia*

ARTICLE HISTORY
Diterima 8 Oktober 2025
Direvisi 10 Desember 2025
Disetujui 16 Desember 2025
Diterbitkan 7 Januari 2026

KATA KUNCI
Apache Spark
Efisiensi energi
Gaussian Mixture Model
k-means
Single Board Computer

KEYWORDS
Apache Spark
Energy efficiency
Gaussian Mixture Model
k-means
Single Board Computer

ABSTRAK. *Penelitian ini bertujuan untuk mengevaluasi kinerja dan efisiensi energi algoritma unsupervised learning terdistribusi pada dua jenis klaster, yaitu Single Board Computer (SBC) dan Personal Computer (PC), menggunakan Apache Spark. Dua algoritma yang diuji, k-means dan Gaussian Mixture Model (GMM), dijalankan pada variasi ukuran dataset dan jumlah inti prosesor untuk mengamati tingkat skalabilitas. Hasil menunjukkan bahwa PC secara konsisten memberikan waktu eksekusi lebih cepat, terutama pada k-means dengan dataset besar. Di sisi lain, SBC menunjukkan efisiensi energi yang lebih tinggi di seluruh skenario, dengan penghematan energi hingga 93% pada k-means dan 86% pada GMM dibandingkan konfigurasi konsumsi tertinggi di PC. Temuan ini menegaskan potensi SBC sebagai solusi komputasi ramah lingkungan (green/sustainable computing) yang hemat daya dan biaya, khususnya untuk pembelajaran, eksperimen akademik, dan pengembangan sistem edge computing skala kecil, serta relevan dalam mendukung agenda keberlanjutan melalui kontribusi terhadap Sustainable Development Goals (SDGs).*

ABSTRACT. *This study aims to evaluate the performance and energy efficiency of distributed unsupervised learning algorithms on two types of clusters, namely Single Board Computers (SBC) and Personal Computers (PC), using Apache Spark. Two algorithms were tested k-means and Gaussian Mixture Model (GMM)–executed across varying dataset sizes and numbers of processor cores to observe scalability. The results show that PCs consistently achieved faster execution times, particularly with k-means on large datasets. On the other hand, SBCs demonstrated higher energy efficiency in all scenarios, with energy savings of up to 93% for k-means and 86% for GMM compared to the highest-consumption configuration on PC. These findings affirm the potential of SBCs as a low-power and cost-efficient solution for green or sustainable computing, particularly for learning, academic experimentation, and small–scale edge computing development, and are relevant to sustainability efforts through their contribution to the Sustainable Development Goals (SDGs).*

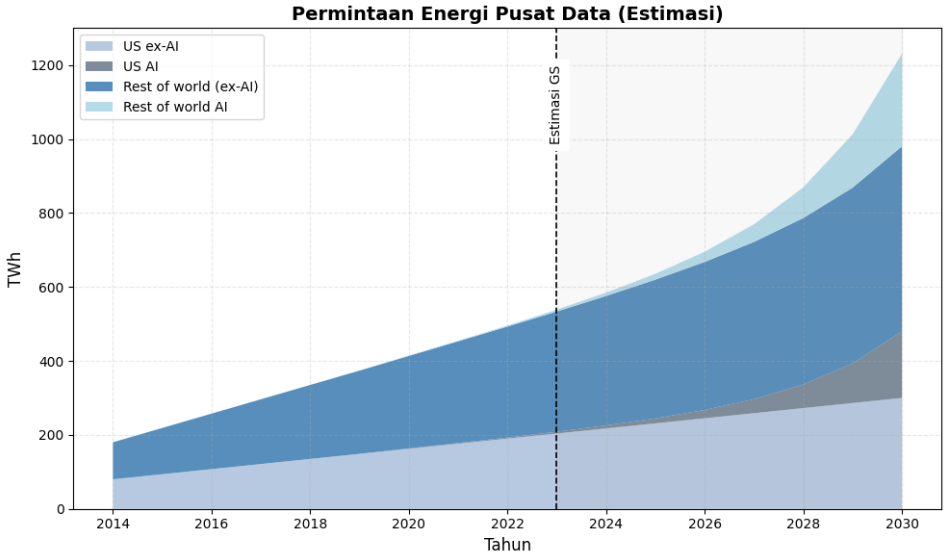
1. Pendahuluan

Dalam satu kali pelatihan, model kecerdasan artifisial (*Artificial Intelligence*/AI) berskala besar seperti ChatGPT dapat mengonsumsi lebih dari 700.000 liter air, atau setara dengan kebutuhan air satu rumah tangga di Amerika Serikat selama 20 tahun [1]. Selain kebutuhan air yang sangat besar tersebut, pusat data global yang mendukung AI dan layanan digital lainnya menyumbang sekitar 1% dari total konsumsi energi dunia [2], dan angka ini diproyeksikan akan terus meningkat seiring dengan pesatnya pertumbuhan teknologi AI. Lebih lanjut, pada tahun 2027, kebutuhan air global untuk sistem AI diperkirakan mencapai 4,2 hingga 6,6 miliar meter kubik, setara dengan konsumsi air tahunan beberapa negara seperti Denmark atau bahkan setengah dari Inggris Raya [3]. Dampak ekologis dari pengembangan AI tidak hanya berasal dari konsumsi energi dan air untuk pelatihan serta pendinginan pusat data, tetapi juga dari proses manufaktur perangkat keras yang menopang sistem AI. Di samping itu, jejak karbon yang dihasilkan dari penggunaan sistem AI secara berkelanjutan, baik dalam pelatihan maupun inferensi, serta dari produksi dan distribusi perangkat keras, kini turut menjadi kontributor utama dalam keseluruhan emisi karbon dari pengembangan AI [4].

Seperti yang ditunjukkan pada Gambar 1, permintaan energi global untuk pusat data, terutama yang digunakan dalam pengembangan AI, diproyeksikan meningkat tajam hingga melampaui 1.000 terawatt-hour (TWh) pada tahun 2030. Proyeksi ini menegaskan bahwa tantangan lingkungan akibat komputasi berskala besar bukan hanya bersifat sementara, melainkan akan terus memburuk seiring dengan eskalasi penggunaan teknologi berbasis data dan AI. Dalam konteks ini, kebutuhan akan solusi komputasi yang efisien dan ramah lingkungan sejalan dengan visi *Green AI* oleh Schwartz et al. [6], yang mendorong evaluasi model tidak hanya berdasarkan akurasi, tetapi juga efisiensi energi. Pendekatan ini mendukung pengembangan AI yang lebih inklusif dan

*Penulis Korespondensi.

Email : *muhazaliebenlito@uinjkt.ac.id* (M. Liebenlito)
Homepage : http://ejurnal.ung.ac.id/index.php/jjom/index / E-ISSN : 2656-1344



Gambar 1. Proyeksi permintaan daya pusat data global hingga 2030, termasuk kontribusi signifikan dari sistem AI [5]

berkelanjutan.

Salah satu pendekatan yang mulai mendapat perhatian di ranah riset dan pendidikan adalah penggunaan SBC. Perangkat ini terbukti cukup kuat untuk menjalankan sistem operasi dan beban komputasi umum, serta dapat dikonfigurasi dalam bentuk klaster yang meniru fitur pusat data dengan biaya dan konsumsi daya yang jauh lebih rendah [7]. Meskipun tidak dirancang untuk skala industri, SBC menawarkan infrastruktur komputasi yang ringkas, efisien, dan terjangkau, terutama dalam eksperimen komputasi paralel dan pemrosesan data berskala menengah.

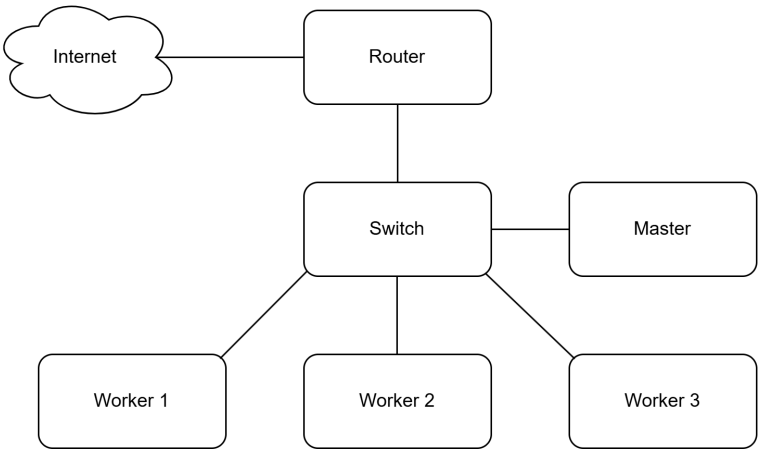
Sejumlah penelitian sebelumnya telah mengeksplorasi pemanfaatan klaster SBC untuk kebutuhan komputasi efisien. Salah satunya adalah studi oleh Bourhnane et al. [8], yang mengimplementasikan platform *edge computing* berbasis klaster SBC (Jetson Nano) dalam aplikasi *smart grid*. Studi tersebut menunjukkan bahwa SBC mampu mengurangi konsumsi energi dan meningkatkan waktu respons dibandingkan dengan platform *cloud* dan PC node tunggal (*single-node* PC), dalam skenario pemrosesan sederhana berbasis Hadoop. Sementara itu, Lee et al. [9] mengevaluasi performa klaster SBC berbasis Raspberry Pi 4B dalam pemrosesan *big data* menggunakan kerangka kerja Hadoop dan Spark. Hasilnya menunjukkan bahwa meskipun performa komputasi SBC relatif terbatas, konsumsi daya yang rendah dan kemudahan pengelolaan menjadikannya pilihan menarik untuk pengolahan data berskala menengah. Namun demikian, kedua studi tersebut belum secara spesifik mengkaji penerapan algoritma *machine learning* secara terdistribusi, khususnya dalam konteks *unsupervised learning*.

Berbeda dari studi sebelumnya, penelitian ini mengangkat konteks *distributed computing* untuk pemrosesan data skala menengah dan pengujian algoritma *machine learning* berbasis *unsupervised learning*, yaitu *k*-means dan GMM. Dengan menggunakan Apache Spark [10] sebagai kerangka kerja komputasi paralel, studi ini mengevaluasi secara langsung performa dan efisiensi energi antara klaster SBC dan PC konvensional dalam konteks eksperimen akademik. Aspek efisiensi energi menjadi perhatian utama dalam studi ini, mengingat minimnya penelitian yang mengukur konsumsi daya selama pelatihan algoritma *unsupervised learning* secara terdistribusi pada perangkat SBC. SBC memiliki keunggulan sebagai solusi komputasi yang hemat biaya dan ramah lingkungan, sehingga relevan untuk mendukung pembelajaran dan eksperimen di lingkungan pendidikan tinggi. Penelitian oleh Qureshi dan Koubaa [11] telah mendemonstrasikan potensi efisiensi energi dari klaster SBC berbasis Hadoop, namun belum mengkaji implementasi algoritma *machine learning* sebagai beban kerja utama. Oleh karena itu, penelitian ini turut berkontribusi dalam pengembangan praktik *green computing* yang hemat energi dan terjangkau untuk pemrosesan data skala menengah.

Untuk itu, eksperimen dilakukan menggunakan kerangka kerja Apache Spark yang diimplementasikan pada dua jenis klaster, yaitu SBC dan PC konvensional, masing-masing terdiri atas tiga node pekerja (*worker nodes*). Apache Spark dipilih karena mendukung pemrosesan data paralel yang efisien dan relatif mudah dikonfigurasi untuk keperluan eksperimen terdistribusi berskala kecil. Evaluasi difokuskan pada dua metrik utama, yaitu waktu eksekusi dan efisiensi energi, saat menjalankan dua algoritma *unsupervised learning*, yakni *k*-means dan GMM. Konfigurasi ini dirancang untuk mencerminkan kebutuhan eksperimen akademik dengan keterbatasan sumber daya, tanpa bertujuan menggantikan sistem komputasi industri berskala besar. Melalui pendekatan ini, studi ini diharapkan dapat memberikan wawasan praktis mengenai pemanfaatan klaster hemat daya dalam konteks pembelajaran, *prototyping*, dan pengembangan sistem terdistribusi berbasis edukasi.

Selain aspek teknis, pendekatan ini juga memiliki relevansi terhadap agenda pembangunan global berkelanjutan. Dengan mengevaluasi efisiensi energi dari perangkat komputasi berdaya rendah seperti SBC, penelitian ini turut berkontribusi dalam mendukung beberapa SDGs, khususnya SDG 7 (Energi Bersih dan Terjangkau), SDG 12 (Konsumsi dan Produksi yang Bertanggung Jawab), serta SDG 13 (Penanganan Perubahan Iklim). Ketiga tujuan ini ditampilkan secara visual dalam Gambar 2, yang menyajikan keseluruhan kerangka SDGs global. Hal ini sejalan dengan laporan SGDs *Report* 2024, yang menekankan pentingnya percepatan transisi menuju sistem energi yang efisien dan ramah lingkungan, termasuk melalui penerapan solusi teknologi rendah emisi dan

Gambar 2. 17 tujuan pembangunan berkelanjutan (*Sustainable Development Goals* – SDGs) [13]



Gambar 3. Arsitektur jaringan klaster SBC dan klaster PC menggunakan topologi *Spark Master–Worker* terhubung melalui *router* dan *switch*

hemat daya sebagai bagian dari transformasi sistemik di sektor teknologi informasi [12].

Hasil dari studi ini diharapkan dapat menjadi dasar eksplorasi lanjutan dalam penerapan *green computing*, khususnya untuk kebutuhan penelitian, pendidikan, dan pengembangan sistem *edge computing* skala kecil. Pendekatan berbasis SBC yang hemat energi juga berpotensi mendukung praktik *green computing* dalam lingkungan akademik, sekaligus mengedukasi pemanfaatan teknologi yang lebih bertanggung jawab secara lingkungan dalam era peningkatan kebutuhan komputasi global.

2. Metode

2.1. Rancangan dan Prosedur Eksperimen

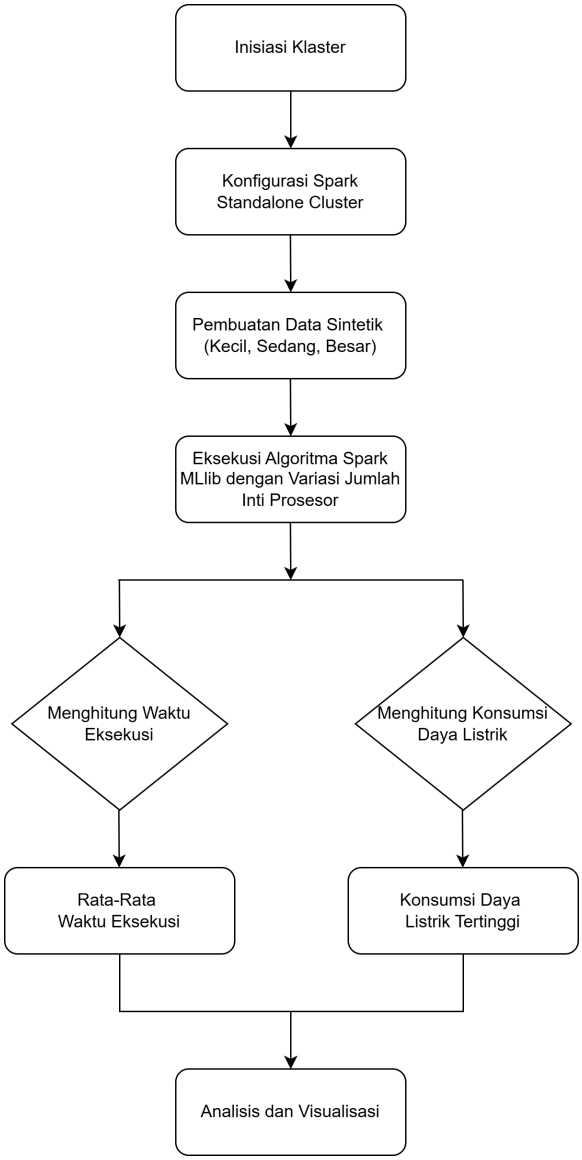
Penelitian ini dirancang sebagai eksperimen komputasi terdistribusi untuk membandingkan kinerja dan efisiensi energi antara dua jenis klaster, yaitu klaster berbasis SBC dan klaster berbasis PC, dalam menjalankan algoritma *unsupervised learning* secara paralel menggunakan Apache Spark, khususnya *k*-means dan GMM. Fokus utama eksperimen ini adalah mengevaluasi waktu eksekusi dan konsumsi daya dari masing-masing klaster dalam konteks eksperimen pendidikan dan pengembangan sistem terdistribusi berskala kecil.

Masing-masing klaster terdiri atas satu node *master* dan tiga node *worker*, sebagaimana ditunjukkan pada Gambar 3. Seluruh node dihubungkan melalui jaringan lokal menggunakan router dan switch, dan dikonfigurasi dalam mode Spark Standalone Cluster. Struktur jaringan ini dipilih karena bersifat sederhana, mudah diimplementasikan, dan cukup representatif untuk eksperimen komputasi paralel dalam lingkungan laboratorium atau pendidikan.

Eksperimen dilakukan melalui serangkaian tahapan sistematis. Tahap pertama adalah inisiasi klaster, yang mencakup instalasi sistem operasi dan konfigurasi jaringan pada masing-masing node sesuai jenis klaster (SBC atau PC). Selanjutnya dilakukan konfigurasi Apache Spark, termasuk penetapan peran *master* dan *worker*, aktivasi mode *standalone cluster*, serta pengaturan jumlah inti prosesor yang dialokasikan ke setiap *worker*. Setelah itu, dibuat dataset sintetik dalam tiga skala ukuran—kecil, sedang, dan besar—untuk menguji skalabilitas sistem.

Pada tahap eksekusi algoritma, program Spark dijalankan untuk mengimplementasikan algoritma *k*-means dan GMM dengan variasi jumlah inti prosesor dan ukuran dataset. Waktu eksekusi diukur berdasarkan selisih waktu sebelum dan sesudah proses komputasi, sedangkan konsumsi daya dicatat menggunakan alat wattmeter eksternal selama proses berlangsung. Setiap konfigurasi digunakan sebanyak lima kali, dengan nilai rata-rata

waktu eksekusi sebagai metrik kinerja, dan nilai konsumsi daya tertinggi dari lima percobaan untuk estimasi konsumsi energi maksimum. Seluruh tahapan ini divisualisasikan dalam Gambar 4, yang menunjukkan alur eksperimental mulai dari inisiasi klaster hingga proses analisis data.



Gambar 4. Diagram alur eksperimen yang menunjukkan tahapan inisiasi klaster, konfigurasi sistem, pembuatan dataset, eksekusi algoritma, dan evaluasi performa serta konsumsi energi

2.2. Dataset

Dataset yang digunakan dalam penelitian ini merupakan data sintetik yang dihasilkan secara acak menggunakan pustaka NumPy, dengan menggunakan nilai awal (*random seed*) untuk menjaga konsistensi dan reproduktibilitas hasil eksperimen. Dataset ini dirancang untuk merepresentasikan karakteristik umum data multivariat pada pengelompokan data, sehingga memungkinkan evaluasi performa sistem komputasi tanpa dipengaruhi oleh kompleksitas data dunia nyata. Dataset yang digunakan dalam eksperimen ini dibagi menjadi tiga skala ukuran sebagai berikut:

- Kecil: 16.384 sampel dengan 20 fitur,
- Sedang: 262.144 sampel dengan 20 fitur, dan
- Besar: 524.288 sampel dengan 20 fitur.

Pembagian ukuran dataset ini bertujuan untuk mengamati bagaimana performa masing-masing klaster dalam menangani beban kerja pada berbagai skala, sekaligus menguji kemampuan sistem terhadap aspek skalabilitas.

2.3. Spesifikasi Perangkat Keras dan Konfigurasi Sistem

Subbagian ini menjelaskan spesifikasi perangkat keras yang digunakan dalam masing-masing klaster, baik SBC maupun PC, untuk mendukung pelaksanaan eksperimen komputasi terdistribusi. Rincian perbandingan disajikan pada Tabel 1.

Kedua klaster dikonfigurasi dengan arsitektur Spark Standalone Cluster, terdiri atas satu *master* node dan tiga *worker* node yang dihubungkan melalui jaringan lokal menggunakan gigabit Ethernet switch (1000 Mbps). Meskipun perangkat *switch* berbeda, keduanya memiliki spesifikasi kecepatan jaringan yang setara. Seluruh node menggunakan sistem operasi Ubuntu Long Term Support (LTS) yang sesuai dengan arsitektur perangkat keras masing-masing, serta menjalankan Apache Spark versi 3.5.3 dengan lingkungan Java berbasis OpenJDK 21.0.4.

Agar proses komputasi dapat berjalan secara terdistribusi dengan optimal, setiap node dikonfigurasi dengan sejumlah parameter. Pengaturan ini mencakup alamat master klaster yang bertugas mengoordinasikan komputasi *(SPARK_MASTER_HOST)*, jumlah inti prosesor yang digunakan untuk pemrosesan *(SPARK_WORKER_CORES)*, serta alokasi memori yang disediakan untuk Spark *(SPARK_WORKER_MEMORY)*. Ketiga parameter ini diatur agar beban kerja dapat dibagi secara seimbang di seluruh node, sehingga mendukung efisiensi pemrosesan pada kedua klaster.

2.4. K-Means

K-means merupakan salah satu algoritma klasterisasi yang cukup populer dalam *unsupervised learning* karena kesederhanaan dan efisiensinya dalam mengelompokkan data berdasarkan kedekatan terhadap titik pusat klaster (*centroid*). Tujuan Algoritma ini adalah meminimalkan total jarak kuadrat antara data dengan *centroid* klaster terdekat. Fungsi objektif yang diminimalkan dalam algoritma k-means dapat dituliskan sebagai berikut:

$$J = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2, \tag{1}$$

di mana:

- k adalah jumlah cluster,
- x_j adalah vektor data ke-j,
- μ_i adalah *centroid* dari cluster ke-i, dan
- C_i adalah himpunan data pada klaster ke-i.

Pada implementasi konvensional, algoritma ini dijalankan secara sekuensial di satu mesin dengan seluruh dataset dimuat ke dalam memori tunggal. Pendekatan ini efektif untuk data berukuran kecil, tetapi menjadi tidak efisien pada data besar karena keterbatasan sumber daya komputasi dan tingginya aktivitas I/O [14]. Untuk mengatasi keterbatasan tersebut, penelitian ini menggunakan implementasi k-means terdistribusi pada kerangka kerja

Tabel 1. Spesifikasi klaster SBC dan klaster PC

Komponen	SBC	PC
Brand	Orange Pi Zero 3	HP Desktop PC M01-F201
Jenis Prosesor	Allwinner H618 ARM Cortex-A53	Intel Core i5-12400 (x86)
Jumlah *Worker Node*	3	3
Jumlah Prosesor (per *node*)	4	12
RAM (per node)	4 GB (LPDDR4)	8 GB (DDR4)
Kecepatan Jaringan	1000 Mbps	1000 Mbps
Penyimpanan	MicroSD 64 GB	SSD 512 GB
Sistem Operasi	Ubuntu 22.04.4 LTS	Ubuntu 24.04.1 LTS

Tabel 2. Perbandingan karakteristik implementasi *k*-means konvensional dan terdistribusi (*Apache Spark*)

Aspek	*k*-means Konvensional	*k*-means Terdistribusi (Apache Spark)
Lokasi Data	Seluruh data tersimpan di satu komputer; terbatas oleh kapasitas memori.	Data dipartisi secara otomatis ke seluruh node dalam klaster (via RDD/*DataFrame*).
Cara Pemrosesan	Dilakukan secara sekuensial oleh satu inti CPU.	Dilakukan secara paralel oleh banyak *worker* node.
Penugasan Data	Jarak semua data ke setiap *centroid* dihitung oleh satu inti CPU.	Setiap *worker* node menghitung jarak dan menetapkan klaster untuk datanya.
Pembaruan Centroid	Dihitung sebagai rata-rata global oleh satu inti CPU.	Tiap *worker* node menghitung *partial sums* dan *counts*; hasil digabung oleh *master* node.
Komunikasi Data	Tidak ada transfer antar node	*Centroid* disiarkan ke *worker*; hasil lokal dikirim ke *master* node.
Skalabilitas	Terbatas oleh kapasitas memori dan kemampuan komputasi mesin tunggal.	Dapat ditingkatkan skalabilitasnya melalui penambahan node.
Toleransi Kesalahan	Rentan terhadap kegagalan komponen tunggal.	Mendukung toleransi kesalahan (*fault tolerance*)
Aplikasi	Dataset kecil hingga menengah.	Analitik data besar (*big data analytics*), pemrosesan waktu nyata (*real-time*).

Apache Spark yang memungkinkan eksekusi paralel di seluruh node klaster. Data direpresentasikan sebagai *Resilient Distributed Datasets* (RDDs) yang secara otomatis dipartisi dan diproses oleh worker node untuk mendukung komputasi paralel dan toleransi kesalahan. Penelitian sebelumnya menunjukan bahwa pemanfaatan Spark secara terdistribusi dapat meningkatkan efiseiensi komputasi k-means pada data berskala besar [15], sehingga pendekatan ini relevan untuk eksperimen komputasi paralel dalam penelitian ini.

Implementasi *k*-means terdistribusi dalam penelitian ini mengikuti enam tahapan utama pada Apache Spark. Proses dimulai dengan inisialisasi *centroid* secara paralel untuk mendapatkan titik awal representatif, diikuti dengan penyiaran (*broadcast*) *centroid* dari *master* node ke seluruh *worker* node untuk memastikan sinkronisasi pusat klaster. Selanjutnya, setiap worker node memproses partisi datanya pada tahap penugasan data (*map phase*) dan menghitung jarak setiap titik terhadap *centroid*. Hasil perhitungan tersebut kemudian diagregasikan secara lokal (*reduce phase*) sebelum dilakukan pembaruan *centroid global* oleh *master* node. Proses ini diulang hingga memenuhi kriteria konvergensi, yaitu ketika perubahan *centroid* sudah tidak signifikan atau jumlah iterasi maksimum tercapai. Dengan pendekatan ini, algoritma *k*-means dapat dijalankan secara efisien dan skalabel, bahkan pada dataset yang melebihi kapasitas memori mesin tunggal. Spark membagi beban komputasi ke seluruh node, sehingga mengurangi ketergantungan pada pemrosesan terpusat. Tabel 2 berikut menyajikan perbandingan karakteristik antara implementasi *k*-means konvensional dan terdistribusi.

Dalam paper ini, pendekatan *k*-means terdistribusi digunakan untuk membandingkan performa dan konsumsi energi pada dua jenis klaster dengan arsitektur perangkat keras berbeda, yaitu SBC dan PC. Gambar 5 berikut menyajikan alur proses *k*-means dalam lingkungan Spark.

2.5. *Gaussian Mixture Model*

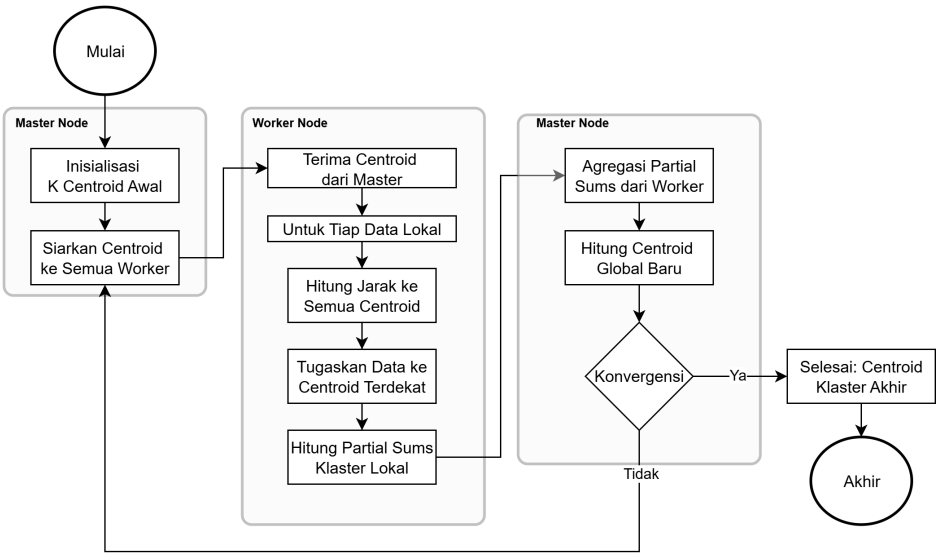
GMM merupakan metode klasterisasi berbasis model probabilistik yang mengasumsikan bahwa data berasal dari kombinasi beberapa distribusi Gaussian. Tidak seperti *k*-means yang menerapkan *hard clustering*, GMM menggunakan pendekatan *soft clustering*, di mana setiap titik data diberi probabilitas keanggotaan terhadap masing-masing klaster. Model ini dilatih menggunakan algoritma Expectation-Maximization (EM) untuk memaksimalkan fungsi log-likelihood dari parameter model [16], sebagaimana ditunjukkan pada Pers. (2):

$$\log P(X|\theta) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \cdot N(x_n|\mu_k, \Sigma_k) \right), \qquad (2)$$

dengan:

- π_k adalah bobot dari distribusi Gaussian ke-k,
- μ_k dan Σ_k adalah *mean* dan kovarian dari distribusi Gaussian ke-k,
- $N(x_n|\mu_k, \Sigma_k)$ adalah fungsi densitas normal multivariat untuk data x_n.

Meskipun prinsip algoritmanya tetap sama, efisiensi implementasi GMM sangat bergantung pada skala data dan arsitektur komputasi. Pada implementasi konvensional, seluruh dataset



Gambar 5. Diagram alur proses algoritma *k*-means terdistribusi dalam *Apache Spark*, dari inisialisasi hingga iterasi konvergen

diproses oleh satu mesin, dan kedua tahap (E-step dan M-step) dieksekusi secara sekuensial. Pendekatan ini cukup untuk data kecil hingga menengah, namun menjadi kendala utama dari segi memori dan waktu ketika dataset menjadi besar. Untuk mengatasi keterbatasan tersebut, GMM dapat diimplementasikan secara terdistribusi menggunakan kerangka kerja Apache Spark, yang memungkinkan pemrosesan paralel terhadap dataset besar. Data dibagi dalam bentuk RDD atau *DataFrame* yang terpartisi secara otomatis ke node-node dalam klaster Spark. Proses EM dalam Spark disusun agar dapat berjalan paralel, dengan pembagian beban kerja di setiap tahap [17].

Implementasi GMM terdistribusi dalam penelitian ini dilakukan menggunakan Apache Spark dengan prinsip Expectation-Maximization (EM). Proses dimulai dengan inisialisasi parameter awal (π_k, μ_k, Σ_k) secara global, kemudian disiarkan (*broadcast*) dari *master* node ke seluruh *worker* node untuk memastikan sinkronisasi model. Selanjutnya, setiap worker node menjalankan tahap E-step untuk menghitung probabilitas posterior (*responsibility*) dari data pada partisi lokalnya, diikuti tahap M-step yang menghasilkan statistik parsial untuk pembaruan parameter global oleh *master* node. Proses ini berulang hingga fungsi log-likelihood mencapai konvergensi atau iterasi maksimum tercapai. Dengan pendekatan ini, GMM dapat diproses secara efisien pada dataset besar tanpa bergantung pada memori tunggal, serta mampu memanfaatkan sumber daya komputasi dari seluruh node dalam klaster. Tabel 3 berikut menyajikan perbandingan antara implementasi GMM konvensional dan terdistribusi menggunakan Spark.

Kemampuan Spark untuk mengatur komputasi paralel GMM menjadi landasan penting dalam penelitian ini untuk membandingkan efisiensi performa dan konsumsi daya antara dua jenis klaster dengan karakteristik berbeda, yaitu SBC dan PC. Gambar 6 berikut menggambarkan diagram alur proses iteratif GMM terdistribusi pada Spark.

2.6. *Evaluasi Performa dan Konsumsi Energi*

Penelitian ini mengevaluasi performa masing-masing klaster dengan mengukur dua aspek utama, yaitu waktu eksekusi (*execution time*) dan konsumsi energi (*energy consumption*) selama proses pelatihan model *unsupervised learning* menggunakan Apache Spark. Untuk setiap konfigurasi sistem (kombinasi jenis klaster, jumlah inti, dan ukuran dataset), proses pelatihan dijalankan sebanyak lima kali. Rata-rata waktu eksekusi dari lima percobaan tersebut diambil sebagai representasi performa komputasi.

Pengukuran konsumsi daya dilakukan menggunakan alat wattmeter digital yang dipasang pada sumber daya (*power supply*) masing-masing klaster. Nilai daya tertinggi yang tercatat selama proses pelatihan dicatat sebagai estimasi beban listrik maksimum. Pencatatan dilakukan secara manual selama program Spark dijalankan dalam mode *Standalone Cluster*. Untuk menghitung total energi listrik yang dikonsumsi selama proses pelatihan, digunakan rumus dasar sebagai berikut:

$$E_{(j)} = P \times t, \tag{3}$$

di mana:

- E = Energi listrik yang dikonsumsi (Joule),
- P = Daya listrik rata-rata (Watt),
- t = Waktu eksekusi (detik).

Jika diperlukan konversi ke dalam satuan watt-jam (Wh) untuk memudahkan interpretasi, digunakan rumus:

$$E_{(Wh)} = \frac{P \times t}{3600}, \tag{4}$$

dengan:

- E = Energi listrik yang dikonsumsi (Wh),
- P = Daya listrik (W),
- t = Waktu penggunaan (detik).

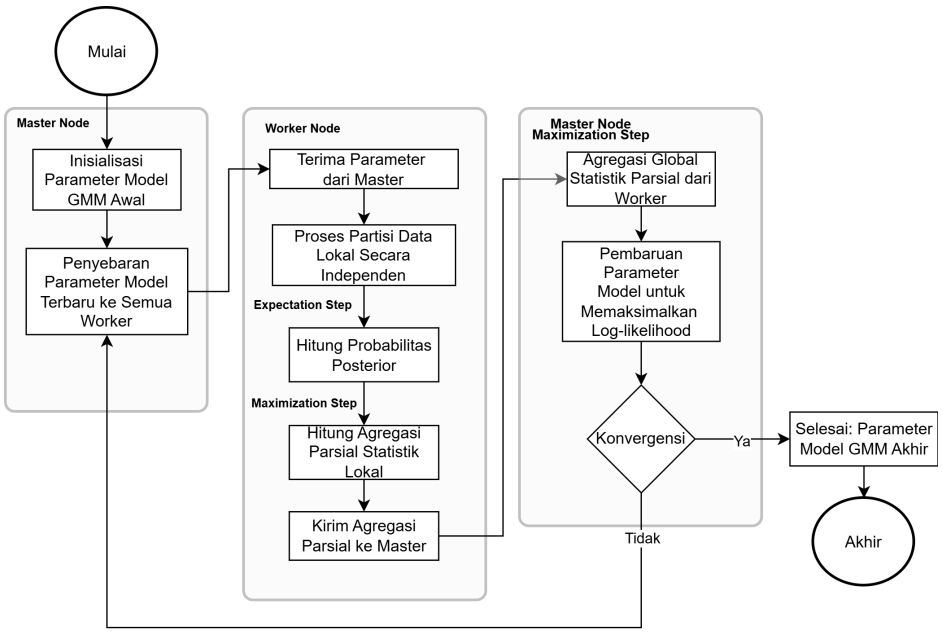
Evaluasi ini bertujuan untuk memberikan gambaran menyeluruh tentang efisiensi komputasi dan konsumsi energi dari masing-masing klaster dalam konteks eksperimen terdistribusi berbasis pendidikan.

3. Hasil dan Pembahasan

Bagian ini menyajikan hasil evaluasi performa klaster SBC dan PC dalam menjalankan algoritma *unsupervised learning* (*k*-means dan GMM) secara terdistribusi menggunakan Apache

Tabel 3. Perbandingan karakteristik implementasi GMM konvensional dan terdistribusi (*Apache Spark*)

Fitur	GMM Konvensional	GMM Terdistribusi (Apache Spark)
Lokasi Data	Seluruh data tersimpan di satu komputer; terbatas oleh kapasitas memori.	Data dipartisi secara otomatis ke seluruh node dalam klaster (via RDD/*DataFrame*).
Paradigma Komputasi	Pemrosesan sekuensial oleh satu inti CPU.	Pemrosesan paralel oleh banyak *worker* node.
Tahap Estep	Perhitungan probabilitas posterior untuk seluruh dataset dilakukan oleh satu inti CPU.	Setiap node *worker* secara paralel menghitung probabilitas posterior untuk data pada partisinya.
Tahap Mstep	Pembaruan parameter model (π_k, μ_k, Σ_k) dilakukan secara global oleh satu inti CPU.	Setiap *worker* node menghitung statistik parsial; hasilnya digabung oleh *master* node.
Komunikasi Data	Tidak ada transfer antar node	Parameter disiarkan (*broadcast*) ke *worker* node; statistik parsial dikirim kembali ke *master* node untuk agregasi.
Skalabilitas	Terbatas oleh kapasitas memori dan kemampuan komputasi mesin tunggal.	Dapat diskalakan secara horizontal melalui penambahan node dalam klaster.
Toleransi Kesalahan	Rentan terhadap kegagalan komponen tunggal.	Memiliki mekanisme toleransi kesalahan.
Aplikasi	Dataset berukuran kecil hingga menengah.	Aplikasi *machine learning* skala besar dan pemrosesan data dengan kebutuhan paralelisme tinggi.



Gambar 6. Diagram alur proses algoritma GMM terdistribusi dalam *Apache Spark*, dari inisialisasi parameter hingga iterasi log-likelihood konvergen

Spark. Evaluasi dilakukan berdasarkan dua metrik utama, yaitu waktu eksekusi dan konsumsi energi, dengan tiga skenario ukuran data: kecil, sedang, dan besar. Seluruh eksperimen dilakukan dengan lima variasi jumlah inti prosesor untuk mengamati pengaruh skalabilitas terhadap performa sistem.

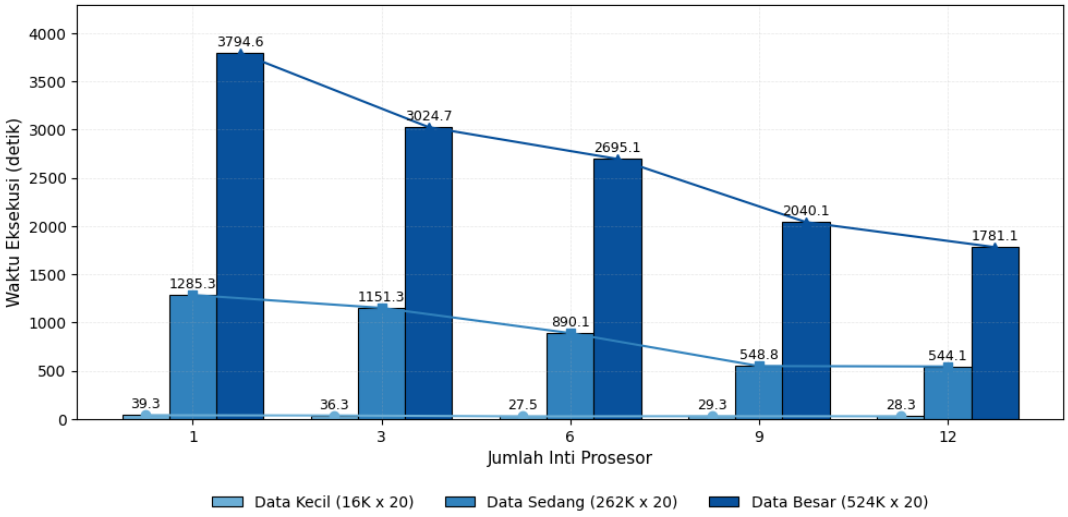
3.1. Waktu Eksekusi

Sub-bab ini membahas hasil evaluasi kinerja algoritma k-means dan GMM pada klaster SBC dan PC, diukur berdasarkan waktu eksekusi total (dalam detik). Analisis akan mencakup dampak variasi jumlah inti prosesor dan skala dataset terhadap kinerja kedua algoritma.

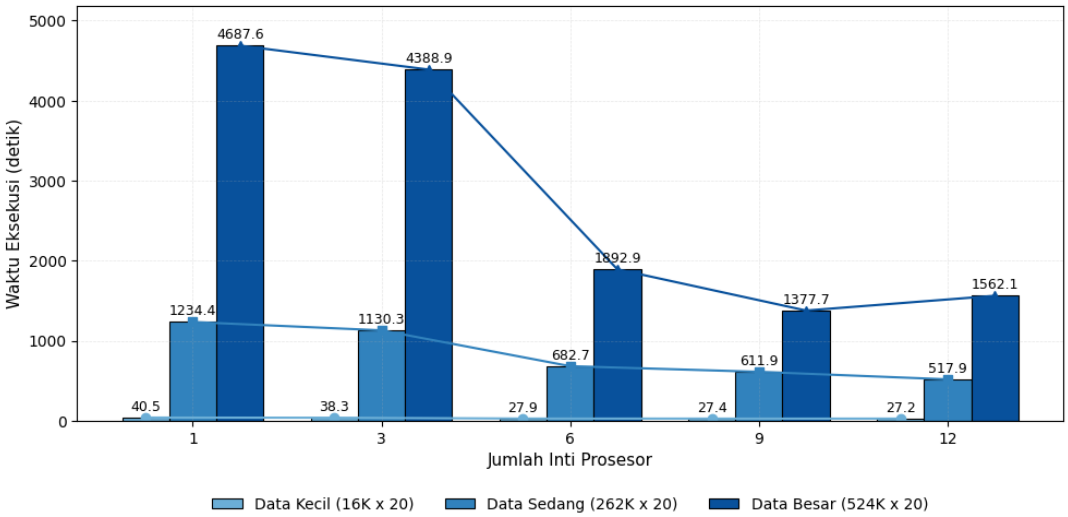
3.1.1. Kinerja Waktu Eksekusi K-Means

Hasil pengukuran waktu eksekusi algoritma *k*-means pada klaster SBC dan PC ditunjukkan pada Gambar 7 dan Gambar 8. Pengujian dilakukan dengan variasi jumlah inti prosesor (1, 3, 6, 9, dan 12) serta tiga skala dataset, yaitu kecil (16.384×20), sedang (262.144×20), dan besar (524.288×20). Secara umum, kedua klaster menunjukkan tren penurunan waktu eksekusi seiring dengan peningkatan jumlah inti prosesor, yang mengindikasikan bahwa pemrosesan paralel menggunakan Apache Spark mampu meningkatkan efisiensi komputasi. Namun, performa antara klaster SBC dan PC menunjukkan perbedaan yang lebih signifikan pada skala dataset yang lebih besar.

Pada klaster SBC, waktu eksekusi untuk dataset kecil tergolong sangat rendah dan stabil, berkisar antara 27 hingga 39 detik di seluruh konfigurasi inti prosesor. Hasil ini menunjukkan bahwa arsitektur terdistribusi Spark tetap efisien untuk dataset berukuran kecil, tanpa menyebabkan beban tambahan yang berarti. Untuk dataset sedang, terjadi penurunan waktu eksekusi dari 1.285,3 detik (satu inti prosesor) menjadi 544,1 detik



Gambar 7. Waktu Eksekusi *k*-means pada klaster SBC



Gambar 8. Waktu eksekusi *k*-means pada klaster PC

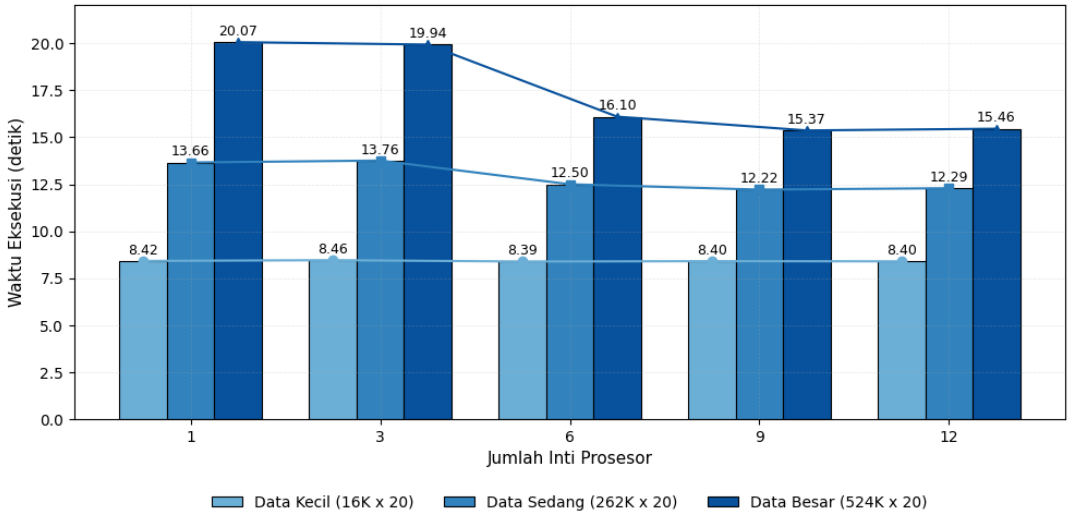
(dua belas inti prosesor), atau mengalami peningkatan efisiensi sebesar 57,7%. Sementara itu, untuk dataset besar, waktu eksekusi menurun dari 3.794,6 detik menjadi 1.781,1 detik, atau setara dengan efisiensi 52,7%. Penurunan waktu eksekusi mulai melambat setelah penggunaan sembilan inti prosesor, yang kemungkinan disebabkan oleh faktor keterbatasan perangkat keras SBC, seperti memori, jaringan antar-node yang lambat, atau proses koordinasi paralel yang semakin berat.

Sementara itu, klaster PC menunjukkan efisiensi pemrosesan yang lebih tinggi, khususnya pada dataset berskala besar. Untuk dataset kecil, waktu eksekusi tetap rendah, yakni antara 27 hingga 40 detik, serupa dengan klaster SBC. Namun, pada dataset sedang, waktu eksekusi menurun dari 1.234,4 detik menjadi 517,9 detik, atau meningkat efisiensinya sebesar 58,0%. Sedangkan untuk dataset besar, penurunan lebih signifikan, dari 4.687,6 detik (satu inti prosesor) menjadi 1.562,1 detik (dua belas inti prosesor), dengan efisiensi mencapai 66,7%. Hasil ini menunjukkan bahwa klaster PC, dalam konteks eksperimen ini, memiliki kapabilitas perangkat keras yang lebih baik dalam menangani beban kerja komputasi-komputasi besar, baik dari sisi prosesor, memori, maupun sistem I/O.

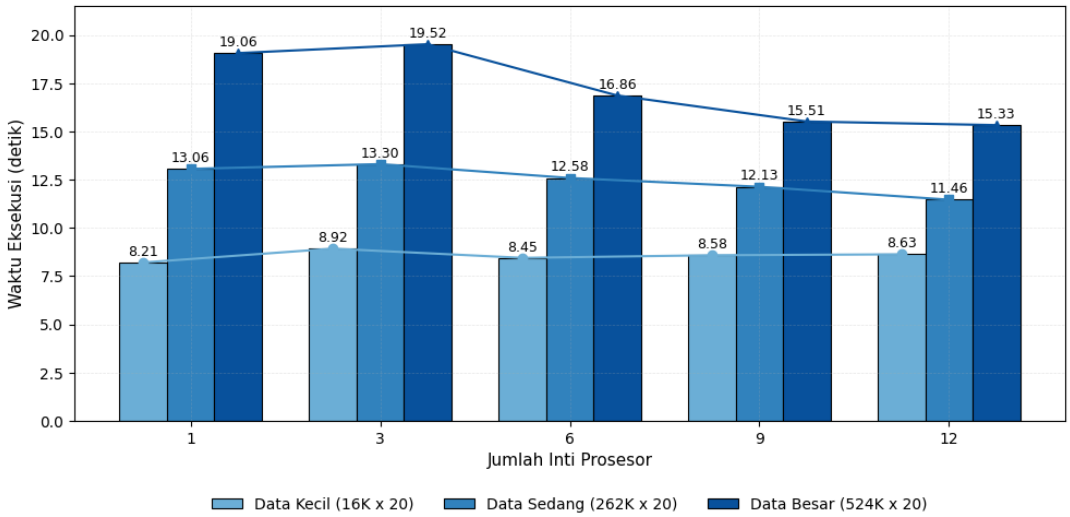
Secara keseluruhan, meskipun klaster SBC menunjukkan performa eksekusi yang lebih rendah dibanding PC, khususnya pada dataset besar, waktu pemrosesan yang dihasilkan masih tergolong layak untuk kebutuhan eksperimen akademik berskala kecil hingga menengah. Dengan mempertimbangkan keterbatasan perangkat kerasnya, SBC tetap menjadi alternatif yang menjanjikan untuk pengajaran dan prototyping sistem komputasi terdistribusi.

3.1.2. Kinerja Waktu Eksekusi Gaussian GMM

Evaluasi performa algoritma GMM ditunjukkan pada Gambar 9 dan Gambar 10, yang menampilkan hasil pengujian pada klaster SBC dan PC dengan variasi jumlah inti prosesor (1, 3, 6, 9, dan 12) serta tiga ukuran dataset yang sama seperti pada eksperimen *k*-means sebelumnya. Berbeda dengan *k*-means, peningkatan jumlah inti prosesor tidak selalu berdampak signifikan terhadap penurunan waktu eksekusi pada GMM. Hal ini disebabkan oleh kompleksitas komputasi probabilistik pada GMM yang melibatkan proses iteratif EM, yang yang cenderung kurang efisien



Gambar 9. Waktu eksekusi GMM pada klaster SBC

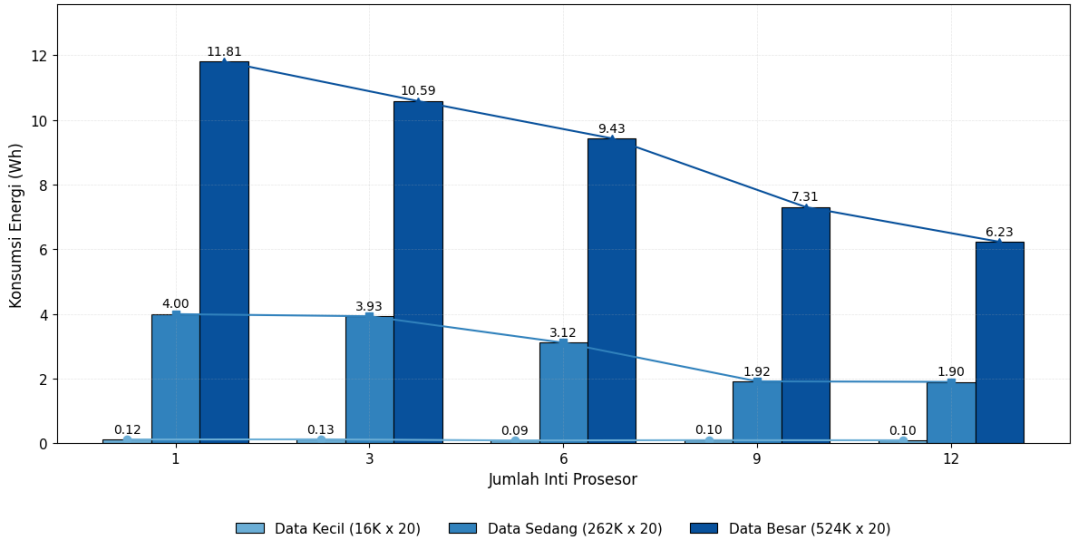


Gambar 10. Waktu eksekusi GMM pada klaster PC

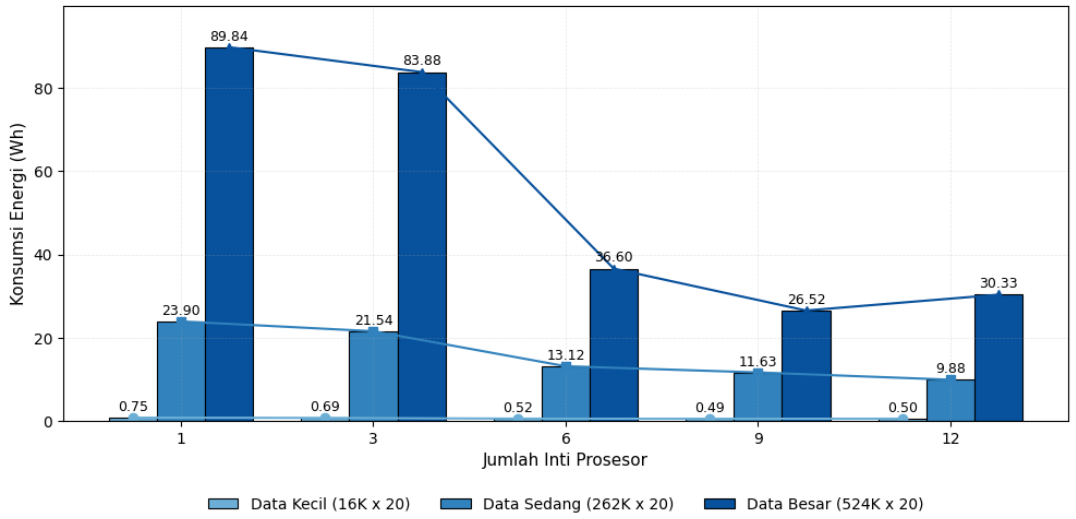
terhadap paralelisasi, terutama pada lingkungan dengan sumber daya komputasi yang terbatas, dibanding metode berbasis *centroid* seperti *k*-means.

Pada klaster SBC, untuk dataset kecil, waktu eksekusi stabil di seluruh konfigurasi inti prosesor, berkisar antara 8,39 hingga 8,46 detik, sehingga peningkatan inti prosesor hampir tidak mempengaruhi waktu eksekusi pelatihan. Untuk dataset sedang, terjadi sedikit penurunan dari 13,66 detik (satu inti prosesor) menjadi 12,29 detik (dua belas inti prosesor), yang hanya mencerminkan efisiensi sekitar 10,0%. Sementara untuk dataset besar, waktu eksekusi justru mengalami fluktuasi, dari 20,07 detik (satu inti prosesor) turun ke 16,10 detik (enam inti prosesor), namun kembali naik menjadi 15,46 detik (dua belas inti prosesor). Penurunan tertinggi hanya sekitar 19,6%, namun tidak konsisten, kemungkinan disebabkan oleh overhead distribusi data dan komunikasi antar-node yang tidak sebanding dengan ukuran beban kerja GMM.

Sementara itu, klaster PC menunjukkan pola yang sedikit lebih baik. Dataset kecil tetap stabil, berkisar antara 8,21 hingga 8,63 detik. Untuk dataset sedang, terjadi penurunan dari 13,06 detik (satu inti prosesor) menjadi 11,46 detik (dua belas inti prosesor), dengan efisiensi sekitar 12,3%. Pada dataset besar, waktu eksekusi menurun dari 19,06 detik menjadi 15,33 detik, atau mengalami efisiensi sebesar 19,6%. Namun, sebagai mana pada klaster SBC, tren penurunan cenderung stagnan setelah penggunaan enam inti prosesor. Hal ini menunjukkan bahwa meskipun PC memiliki spesifikasi perangkat keras yang lebih tinggi, performa GMM tidak meningkat secara linear seiring penambahan jumlah inti, karena beban komputasi GMM lebih kompleks dibandingkan *k*-means. Secara umum, baik SBC maupun PC menunjukkan bahwa GMM tidak terlalu diuntungkan dari paralelisasi berbasis Spark, terutama pada dataset kecil dan sedang. Performa cenderung stabil dan kurang responsif terhadap peningkatan jumlah inti prosesor, berbeda dengan *k*-means yang skalabilitasnya lebih linear. Hal ini penting untuk dicatat bahwa tidak semua algoritma machine learning dapat memanfaatkan arsitektur terdistribusi secara optimal, terutama untuk model probabilistik seperti GMM. Kendati demikian, waktu eksekusi yang relatif singkat (di bawah 25 detik) tetap menunjukkan bahwa kedua jenis klaster cukup mampu menangani kebutuhan komputasi GMM dalam



Gambar 11. Konsumsi energi *k*-means pada klaster SBC



Gambar 12. Konsumsi energi *k*-means pada klaster PC

konteks eksperimen akademik.

3.2. Konsumsi Energi

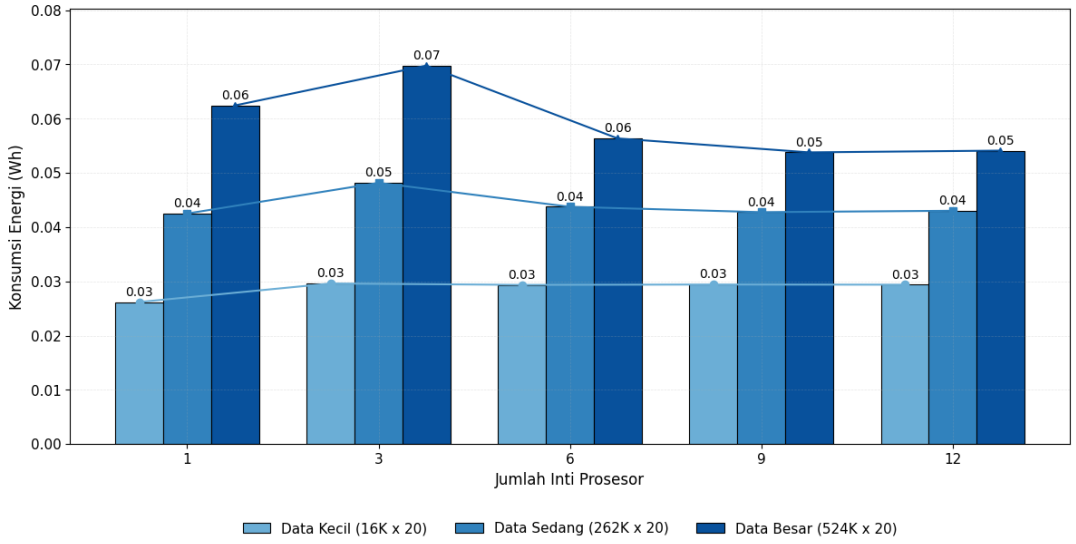
Sub-bab ini menganalisis konsumsi energi total (dalam Watt-jam, Wh) untuk algoritma *k*-means dan GMM yang dijalankan pada klaster SBC dan PC. Analisis akan fokus pada dampak variasi jumlah inti prosesor dan ukuran dataset terhadap efisiensi energi.

3.2.1. Konsumsi Energi K-Means

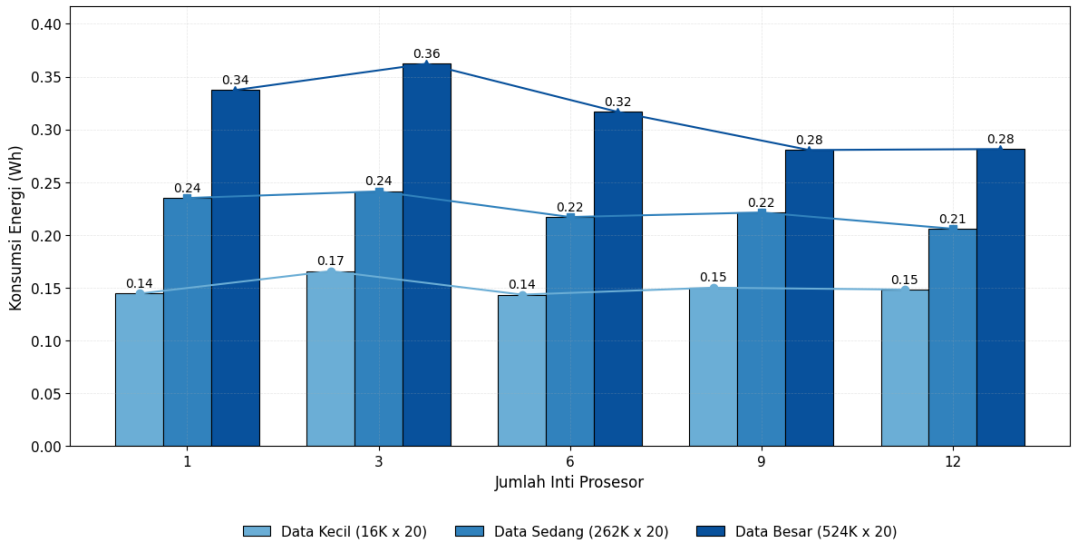
Konsumsi energi selama eksekusi algoritma *k*-means ditunjukkan pada Gambar 11 dan Gambar 12 untuk masing-masing klaster SBC dan PC. Pengukuran dilakukan menggunakan wattmeter eksternal dan dinyatakan dalam satuan Wh berdasarkan berbagai variasi jumlah inti prosesor dan ukuran dataset. Hasil menunjukkan bahwa peningkatan jumlah inti prosesor umumnya berkontribusi terhadap penurunan konsumsi energi total, terutama pada dataset berukuran sedang dan besar. Hal ini berkorelasi dengan waktu eksekusi yang lebih singkat pada konfigurasi dengan lebih banyak inti prosesor.

Pada klaster SBC, konsumsi energi tergolong sangat rendah di semua skenario. Untuk dataset kecil, konsumsi relatif stabil, yaitu antara 0,09 hingga 0,13 Wh. Pada dataset sedang, konsumsi menurun dari 4,00 Wh (satu inti prosesor) menjadi 1,90 Wh (dua belas inti prosesor), mencerminkan efisiensi energi sebesar 52,5%. Efisiensi serupa terlihat pada dataset besar, di mana konsumsi energi menurun dari 11,81 Wh menjadi 6,23 Wh, atau sebesar 47,3%. Penurunan ini menunjukkan bahwa eksekusi paralel secara efektif mengurangi waktu proses dan beban daya, meskipun skalabilitasnya mulai melandai pada jumlah inti prosesor tinggi.

Berbeda dengan SBC, klaster PC menunjukkan pola serupa dengan konsumsi energi yang jauh lebih tinggi. Pada dataset kecil, konsumsi stabil dalam rentang 0,49 hingga 0,75 Wh. Untuk dataset sedang, konsumsi energi menurun dari 23,90 Wh (satu inti prosesor) menjadi 9,88 Wh (dua belas inti prosesor), dengan penghematan sebesar 58,7%. Penurunan lebih drastis tercatat pada dataset besar, dari 89,84 Wh (satu inti prosesor) menjadi 30,33



Gambar 13. Konsumsi energi GMM pada klaster SBC



Gambar 14. Konsumsi energi GMM pada klaster PC

Wh (dua belas inti prosesor), yang mencerminkan efisiensi energi sebesar 66,3%. Hal ini menunjukkan bahwa meskipun klaster PC menyerap daya lebih besar, skalabilitasnya cukup baik pada beban kerja besar, sehingga efisiensi energi tetap dapat dicapai melalui peningkatan paralelisasi.

Klaster SBC menunjukkan keunggulan signifikan dari sisi efisiensi energi dibandingkan klaster PC, terutama pada eksperimen ringan hingga menengah. Meskipun performa pemrosesannya lebih rendah dibandingkan PC, daya yang dikonsumsi jauh lebih kecil. Sebagai perbandingan, pada skenario dataset besar dengan konfigurasi dua belas inti prosesor, konsumsi energi SBC tercatat sebesar 6,23 Wh, sedangkan PC mencapai 30,33 Wh—sekitar 79% lebih tinggi. Menariknya, dibandingkan dengan konfigurasi PC paling tinggi konsumsi (satu inti pada dataset besar), penghematan energi oleh SBC mencapai lebih dari 93%. Hasil ini menunjukkan bahwa SBC memiliki keunggulan efisiensi energi yang signifikan dalam konteks eksperimen ini, sehingga layak dipertimbangkan sebagai alternatif hemat daya untuk pengembangan sistem komputasi terdistribusi berbasis *green computing* di lingkungan pendidikan dan penelitian berskala kecil.

3.2.2. Konsumsi Energi GMM

Konsumsi energi selama pelaksanaan algoritma GMM pada klaster SBC dan PC ditampilkan pada Gambar 13 dan Gambar 14. Berbeda dari algoritma *k*-means, konsumsi energi GMM secara umum lebih rendah dan relatif stabil pada berbagai konfigurasi jumlah inti prosesor dan ukuran dataset. Hal ini mencerminkan bahwa GMM, dalam konteks eksperimen ini, memiliki waktu eksekusi yang singkat serta beban pemrosesan yang tidak terlalu intensif. Meski demikian, masih terlihat perbedaan tingkat efisiensi antara kedua jenis klaster.

Pada klaster SBC, konsumsi daya tercatat sangat rendah. Untuk dataset kecil, konsumsi energi konstan di angka 0,03 Wh di seluruh konfigurasi inti prosesor. Pada dataset sedang, terjadi sedikit penurunan dari 0,05 Wh (tiga inti prosesor) menjadi 0,04 Wh (dua belas inti prosesor), yang setara dengan efisiensi sekitar 20%. Sementara untuk dataset besar, konsumsi menurun dari 0,07 Wh (tiga inti prosesor) menjadi 0,05 Wh (dua belas inti pro-

sesor) dalam rentang yang sama, mencerminkan efisiensi energi sebesar 28,6%. Meskipun skalanya kecil, tren ini tetap menunjukkan bahwa peningkatan jumlah inti prosesor dapat membantu mengurangi konsumsi daya secara moderat dalam eksekusi GMM di klaster SBC.

Pada klaster PC, pola konsumsi energi juga menunjukkan hal yang serupa namun dengan angka konsumsi yang lebih tinggi. Dataset kecil berada dalam rentang 0,14 hingga 0,17 Wh, sementara dataset sedang menunjukkan penurunan dari 0,24 Wh (1 hingga 3 inti prosesor) menjadi 0,21 Wh (dua belas inti prosesor), dengan efisiensi sekitar 12,5%. Untuk dataset besar, konsumsi energi menurun dari 0,36 Wh (tiga inti prosesor) menjadi 0,28 Wh (dua belas inti prosesor), memberikan efisiensi sekitar 22,2%. Penurunan ini menunjukkan bahwa penambahan inti prosesor memberikan pengaruh terbatas terhadap efisiensi daya pada algoritma GMM, terutama jika dibandingkan dengan algoritma *k*-means.

Secara keseluruhan, baik SBC maupun PC menunjukkan bahwa konsumsi energi untuk GMM jauh lebih rendah dibandingkan *k*-means. Namun, SBC secara konsisten lebih hemat energi. Sebagai ilustrasi, pada skenario dataset besar dengan konfigurasi dua belas inti prosesor, konsumsi energi SBC tercatat sebesar 0,05 Wh, sedangkan PC mencapai 0,28 Wh—yang menunjukkan perbedaan sebesar 82%. Lebih lanjut, jika dibandingkan dengan konfigurasi konsumsi tertinggi pada klaster PC (0,36 Wh), penghematan energi oleh SBC bahkan mencapai lebih dari 86%. Dengan konsumsi energi yang sangat rendah dan performa yang cukup dalam eksperimen akademik, SBC tetap menjadi platform menarik untuk implementasi algoritma *machine learning* ringan dan pengembangan sistem berbasis *green computing* di lingkungan pendidikan.

4. Kesimpulan

Penelitian ini menunjukkan bahwa tantangan konsumsi energi dan dampak lingkungan dari sistem AI yang semakin meluas dapat direspons dengan pendekatan *green computing*, khususnya dengan memanfaatan klaster SBC yang hemat daya. Eksperimen menggunakan Apache Spark untuk menjalankan algoritma *unsupervised learning* (*k*-means dan GMM) secara terdistribusi membuktikan bahwa meskipun klaster PC memberikan waktu eksekusi lebih cepat terutama pada dataset besar, klaster SBC secara konsisten menawarkan efisiensi energi yang signifikan—dengan penghematan energi hingga 93% pada *k*-means dan 86% pada GMM dibandingkan konfigurasi PC konsumsi tertinggi. Hal ini menunjukkan bahwa SBC berpotensi menjadi solusi terjangkau dan ramah lingkungan untuk mendukung kebutuhan eksperimen akademik, *prototyping*, dan pengembangan sistem *edge computing* skala kecil.

Dengan hasil tersebut, penelitian ini menegaskan pentingnya mengevaluasi kinerja komputasi tidak hanya dari segi kecepatan tetapi juga efisiensi energi, sejalan dengan visi Green AI yang mendorong pengembangan sistem lebih berkelanjutan. Temuan ini mendukung upaya pencapaian SDG 7 (Energi Bersih dan Terjangkau), SDG 12 (Konsumsi dan Produksi yang Bertanggung Jawab), serta SDG 13 (Penanganan Perubahan Iklim) dengan menyediakan opsi infrastruktur komputasi rendah emisi untuk pendidikan tinggi. Di masa mendatang, hasil penelitian ini dapat menjadi dasar pengembangan lanjutan dalam penerapan algoritma *machine learning* berbasis Spark di lingkungan SBC untuk mendukung praktik *green computing* di sektor riset, pendidikan, dan pengembangan teknologi informasi berkelanjutan.

Kontribusi Penulis. **Deffin Purnama Noer**: Konseptualisasi, metodologi, perangkat lunak, analisis formal, penulisan—penyusunan draf asli, visualisasi. **Muhaza Liebenlito**: Konseptualisasi, metodologi, penulisan—tinjauan dan penyuntingan, supervisi, validasi. **Taufik Edy Sutanto**: Konseptualisasi, metodologi, penulisan—tinjauan dan penyuntingan, supervisi, validasi. Semua penulis telah membaca dan menyetujui versi akhir naskah yang diterbitkan.

Ucapan Terima Kasih. Para penulis mengucapkan terima kasih kepada editor dan para reviewer atas masukan dan saran berharga yang telah berkontribusi terhadap penyempurnaan penelitian ini serta perbaikan manuskrip.

Pembiayaan. Penelitian ini tidak menerima pendanaan eksternal.

Konflik Kepentingan. Para penulis menyatakan tidak ada konflik kepentingan yang terkait dengan artikel ini.

Ketersediaan Data. Tidak tersedia.

Referensi

[1] A. S. George, A. S. H. George, and A. S. G. Martin, "The Environmental Impact of AI: A Case Study of Water Consumption by Chat GPT," *Partners Universal International Innovation Journal*, vol. 1, no. 2, pp. 97–104, 2023, [Online]. Available: https://puiij.com/index.php/research/article/view/39 [Accessed: 17-June-2025].

[2] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, 2020, doi: 10.1126/science.aba3758.

[3] P. Li, J. Yang, M. A. Islam, and S. Ren, "Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models," apr 2023, [Online]. Available: http://arxiv.org/abs/2304.03271 [Accessed: 19-June-2025].

[4] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. A. Behram, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. S. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood, "Sustainable AI: Environmental Implications, Challenges and Opportunities," oct 2021, [Online]. Available: http://arxiv.org/abs/2111.00364 [Accessed: 22-June-2025].

[5] Goldman Sachs, "AI is poised to drive 160demand," [Online]. Available: https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand [Accessed: 17-June-2025].

[6] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, nov 2020, doi: 10.1145/3381831.

[7] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, 2018, doi: 10.1016/j.future.2018.06.048.

[8] S. Bourhnane, M. R. Abid, K. Zine-Dine, N. Elkamoun, and D. Benhaddou, "Cluster of single-board computers at the edge for smart grids applications," *Applied Sciences (Switzerland)*, vol. 11, no. 22, nov 2021, doi: 10.3390/app112210981.

[9] E. Lee, H. Oh, and D. Park, "Big Data Processing on Single Board Computer Clusters: Exploring Challenges and Possibilities," *IEEE Access*, vol. 9, pp. 142 551–142 565, 2021, doi: 10.1109/ACCESS.2021.3120660.

[10] Apache Software Foundation, "Apache Spark™ - Unified Engine for large-scale data analytics," [Online]. Available: https://spark.apache.org/ [Accessed: 19-June-2025].

[11] B. Qureshi and A. Koubaa, "On energy efficiency and performance evaluation of single board computer based clusters: A hadoop case study," *Electronics (Switzerland)*, vol. 8, no. 2, 2019, doi: 10.3390/electronics8020182.

[12] United Nations, "Progress towards the Sustainable Development Goals:

Report of the Secretary-General 2024," https://sdgs.un.org/goals, 2024, [Online]. Available: https://sdgs.un.org/goals [Accessed: 20-June-2025].

[13] ——, "THE 17 GOALS | Sustainable Development," https://sdgs.un.org/goals, [Online]. Available: https://sdgs.un.org/goals [Accessed: 25-June-2025].

[14] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation*, pp. 15–28, 2012, [Online]. Available: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf [Accessed: 3-July-2025].

[15] Y. Feng, J. Zou, W. Liu, and F. Lv, "Distributed K-Means algorithm based on a Spark optimization sample," *PLoS ONE*, vol. 19, no. 12, pp. 1–21, 2024, doi: 10.1371/journal.pone.0308993.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977, doi: 10.1111/j.2517-6161.1977.tb01600.x.

[17] A. A. Ratnaparkhi, E. Pilli, and R. C. Joshi, "Scaling GMM Expectation Maximization algorithm using bulk synchronous Parallel approach," *Proceedings of the 2015 International Conference on Green Computing and Internet of Things, ICGCIoT 2015*, pp. 558–562, 2016, doi: 10.1109/ICGCIoT.2015.7380527.