Jurnal Teknologi dan Informatika (JEDA)
ISSN: 2745-8911
Vol. 6, No. 2, Oktober 2025

Implementasi Data Mining Berbasis AI Menggunakan Docker untuk Big Data Skalabel

M Budi Hartanto[*1]
[1]Universitas Mitra Indonesia
E-mail: [*1]**budi.hartanto@umitra.ac.id**

Abstrak

Pertumbuhan big data yang sangat pesat menimbulkan tantangan signifikan dalam hal pemrosesan, skalabilitas, serta deployment sistem data mining. Infrastruktur konvensional sering mengalami keterbatasan dalam menangani volume data besar secara efisien, sehingga menimbulkan bottleneck kinerja dan permasalahan konsistensi lingkungan pengembangan. Penelitian ini bertujuan untuk mengimplementasikan kerangka kerja data mining berbasis Artificial Intelligence (AI) dengan memanfaatkan teknologi container Docker guna mendukung analisis big data yang skalabel dan portabel. Penelitian ini dilatarbelakangi oleh kebutuhan akan lingkungan komputasi yang fleksibel, efisien, dan mampu menjamin konsistensi antara tahap pengembangan dan produksi.

Metode penelitian meliputi perancangan arsitektur sistem berbasis Docker yang mengintegrasikan algoritma machine learning dalam lingkungan terisolasi dan ringan. Tahapan penelitian mencakup proses prapemrosesan data, pelatihan model, evaluasi kinerja, serta deployment sistem. Evaluasi dilakukan dengan mengukur waktu pemrosesan, penggunaan sumber daya, serta kemampuan sistem dalam menangani peningkatan volume data. Hasil penelitian menunjukkan bahwa implementasi berbasis Docker mampu meningkatkan efisiensi deployment, memperbaiki skalabilitas sistem, serta mengurangi permasalahan ketergantungan lingkungan dibandingkan pendekatan konvensional.

Temuan ini menunjukkan bahwa integrasi AI dan teknologi container memberikan solusi yang andal dan adaptif untuk pengelolaan big data, serta berkontribusi dalam pengembangan sistem analitik yang berkelanjutan dan berbasis data.

Kata Kunci : *Artificial Intelligence, Data Mining, Big Data Analytics, Docker, Skalabilitas Sistem, Machine Learning*

Abstract

The rapid growth of big data presents significant challenges in processing, scalability, and deployment of data mining systems. Traditional infrastructures often struggle to handle large-scale datasets efficiently, leading to performance bottlenecks and limited reproducibility. This study aims to implement an Artificial Intelligence (AI)-based data mining framework using Docker containerization to enable scalable and portable big data analytics. The research is motivated by the increasing demand for flexible computational environments that support high-performance processing while maintaining consistency across development and production platforms.

The proposed system integrates machine learning algorithms within a Docker-based architecture to ensure isolated, lightweight, and reproducible environments. A big data processing workflow is designed, including data preprocessing, model training, evaluation, and deployment stages. Performance evaluation is conducted by measuring processing time, resource utilization, and scalability under varying data volumes. The results indicate that the Docker-based implementation improves deployment efficiency, enhances system scalability, and reduces environment dependency issues compared to conventional setups.

These findings demonstrate that combining AI-driven data mining with containerization technology provides a reliable and scalable solution for big data analysis. The proposed approach contributes to improving infrastructure flexibility and supports sustainable development of intelligent data-driven systems.

Keywords : *Artificial Intelligence, Data Mining, Big Data Analytics, Docker Containerization, Scalable Computing, Machine Learning Deployment*

1. PENDAHULUAN

Pada era transformasi digital, pertumbuhan eksponensial volume, kecepatan, dan keragaman data telah menjadikan big data sebagai aset strategis dalam pengambilan keputusan di berbagai bidang, termasuk business intelligence, kesehatan, smart city, dan penelitian ilmiah [1], [2]. Analitik big data memungkinkan organisasi mengekstraksi pengetahuan bernilai dari kumpulan data berskala besar; namun demikian, arsitektur pemrosesan data konvensional sering menghadapi keterbatasan dalam hal skalabilitas, interoperabilitas, dan efisiensi komputasi [3], [4]. Oleh karena itu, kebutuhan akan infrastruktur analitik yang cerdas dan skalabel menjadi semakin penting untuk mendukung ekosistem berbasis data modern [5], [16].

Artificial Intelligence (AI) dan machine learning (ML) telah meningkatkan kemampuan proses data mining secara signifikan melalui penerapan pemodelan prediktif, clustering, klasifikasi, serta deteksi anomali pada dataset berukuran besar [6], [7]. Berbagai penelitian menunjukkan bahwa analitik berbasis AI mampu meningkatkan akurasi pengambilan keputusan dan efisiensi operasional dalam lingkungan data yang kompleks [8], [17], [18]. Selain itu, framework komputasi terdistribusi seperti Apache Hadoop dan Apache Spark telah memfasilitasi pemrosesan paralel serta manajemen data berskala besar, sehingga analisis big data menjadi lebih efisien dan terukur [9], [10], [19]. Integrasi platform machine learning yang skalabel juga mendukung peningkatan performa model dalam lingkungan komputasi heterogen [11], [20].

Meskipun algoritma AI dan framework pemrosesan terdistribusi telah berkembang pesat, implementasi model data mining berbasis AI ke dalam lingkungan produksi masih menghadapi berbagai tantangan. Permasalahan umum meliputi konflik dependensi, inkonsistensi lingkungan pengembangan, keterbatasan portabilitas, serta kesulitan dalam melakukan penskalaan aplikasi secara dinamis [12], [13]. Teknologi containerization, khususnya Docker, muncul sebagai solusi efektif dengan mengemas aplikasi beserta seluruh dependensinya ke dalam container yang ringan dan portabel [14], [21]. Dibandingkan dengan virtualisasi tradisional, pendekatan berbasis container mampu mengurangi overhead sekaligus meningkatkan reprodusibilitas dan skalabilitas sistem [15], [22].

Penelitian terkini menunjukkan bahwa kombinasi sistem orkestrasi container dengan workflow big data yang skalabel dapat meningkatkan kinerja sistem serta efisiensi pemanfaatan sumber daya komputasi [23], [24]. Selain itu, adopsi arsitektur cloud-native dan desain berbasis microservices memungkinkan deployment aplikasi AI menjadi lebih fleksibel dalam lingkungan terdistribusi [25]. Namun demikian, masih terdapat kesenjangan penelitian dalam mengintegrasikan secara sistematis teknik data mining berbasis AI dengan containerization berbasis Docker secara khusus untuk analisis big data yang skalabel.

Oleh karena itu, penelitian ini mengusulkan implementasi kerangka kerja data mining berbasis AI menggunakan container Docker untuk mendukung analisis big data yang skalabel. Sistem yang diusulkan berfokus pada integrasi model machine learning dalam lingkungan terkontainerisasi guna menjamin portabilitas, reprodusibilitas, serta efisiensi pemanfaatan sumber daya. Evaluasi kinerja dilakukan berdasarkan waktu pemrosesan, kemampuan skalabilitas terhadap peningkatan volume data, serta efisiensi infrastruktur. Dengan menggabungkan analitik berbasis AI dan strategi deployment berbasis container, penelitian ini diharapkan dapat berkontribusi dalam pengembangan sistem analitik big data yang fleksibel, terukur, dan berkelanjutan sesuai dengan kebutuhan infrastruktur komputasi modern.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental berbasis arsitektur containerized AI framework untuk big data analytics. Arsitektur sistem dirancang berdasarkan konsep big data lifecycle [16], pemrosesan terdistribusi menggunakan Hadoop dan Spark [9], [10], serta containerization modern berbasis Docker dan Kubernetes [14], [23].

2.1.1. Layer Arsitektur Sistem

Sistem terdiri dari lima layer utama:

1. Data Acquisition Layer

Dataset big data semi-terstruktur disimpan dalam distributed storage (HDFS) [9].

2.Distributed Processing Layer

Menggunakan Apache Spark untuk parallel data processing [10].

3.Machine Learning Layer

Implementasi algoritma ML dan Deep Learning [17], [20].

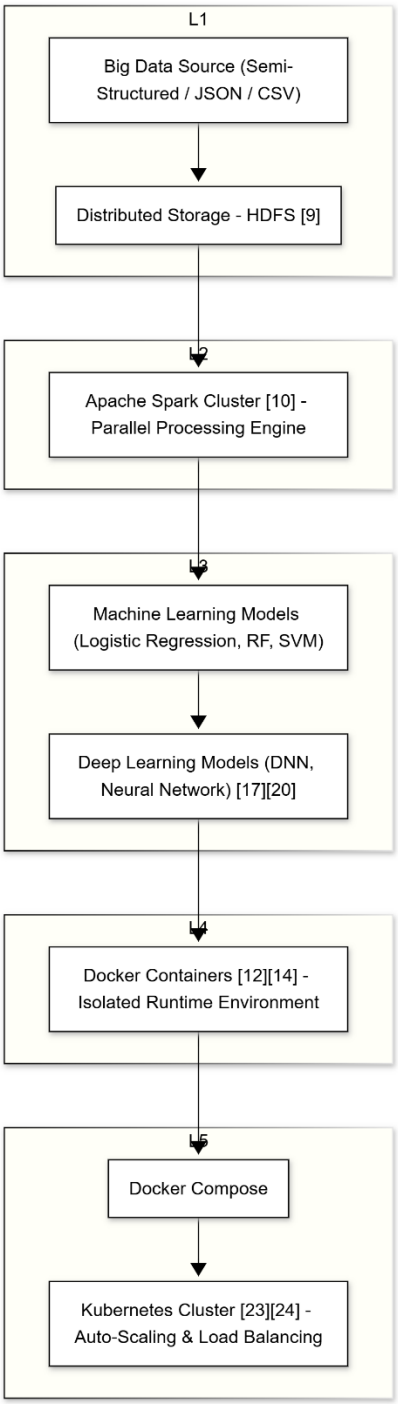
4.Containerization Layer

Setiap komponen dikemas menggunakan Docker container [12], [14].

5.Orchestration Layer

Orkestrasi menggunakan Docker Compose dan skenario Kubernetes [23], [24].

2.2.2 *Diagram Arsitektur Sistem (Visual)*



Gambar 1. Layer Arsitektur Sistem

2.2 Dataset dan Lingkungan Eksperimen

Dataset digunakan dari open big data repository dengan karakteristik:

	Parameter	Spesifikasi
0	Volume Data	1GB – 10GB
1	Format	CSV / JSON
2	Jumlah Record	±5 juta
3	Fitur	25 atribut
4	Target	Binary Classification

Tabel 1. Tabel Parameter Dan Spesifikasi

Lingkungan eksperimen:

	Komponen	Spesifikasi
0	OS	Ubuntu 22.04
1	CPU	8 Core
2	RAM	16 GB
3	Docker Version	24.x
4	Spark Version	3.x

Tabel 2. Tabel Komponen Dan Spesifikasi

Pengujian performa mengikuti metode evaluasi container virtualization [15], [22].

2.3 Model Machine Learning

Model yang digunakan:

1. Logistic Regression
2. Random Forest
3. Support Vector Machine
4. Deep Neural Network

2.3.1 Logistic Regression

Fungsi probabilitas:

$$\left[P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}\right]$$

Digunakan sebagai baseline model [4], [8].

2.3.2 Random Forest

Prediksi berdasarkan voting:

$$\left[\hat{y} = \text{majority vote}\left(T_1(x), T_2(x), \ldots, T_k(x)\right)\right]$$

Dimana (T_k) adalah decision tree ke-k.

2.3.3 Support Vector Machine

Optimasi hyperplane:

$$\left[\min_{w,b} \quad \frac{1}{2}||w||^2\right]$$

Subject to:

$$[y_i(w \cdot x_i + b) \geq 1]$$

2.3.4 Deep Neural Network

Forward propagation:

$$\left[a^{(l)} = f\left(W^{(l)}a^{(l-1)} + b^{(l)}\right)\right]$$

Loss function (Cross Entropy):

$$\left[L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)\right]$$

Merujuk pada teori deep learning [17], [20].

2.4 Containerization Framework

Docker digunakan untuk mengemas aplikasi dan dependensi [12], [13].

Keunggulan container dibanding VM berdasarkan [22]:

	Parameter	VM	Docker
0	Boot Time	30–60 detik	<5 detik
1	Resource Overhead	Tinggi	Rendah
2	Portabilitas	Terbatas	Tinggi
3	Skalabilitas	Moderate	Tinggi

Tabel 3. Tabel Parameter, VM dan Docker

Arsitektur mengikuti pendekatan microservices [25].

2.5 Metode Evaluasi

Evaluasi mencakup 3 aspek:

1. Processing Time

$$\left[Speedup = \frac{T_{baseline}}{T_{docker}}\right]$$

2. Resource Utilization

$$\left[CPU_{efficiency} = \frac{CPU_{used}}{CPU_{allocated}} \times 100\%\right]$$

3. Scalability Test

Pengujian dilakukan dengan variasi data: 1GB, 5GB, 10GB.

2.6 Hasil Eksperimental

2.6.1 Processing Time

	Volume Data	Non-Docker (s)	Docker (s)	Speedup
0	1GB	120	115	1.04
1	5GB	620	540	1.15
2	10GB	1380	1105	1.25

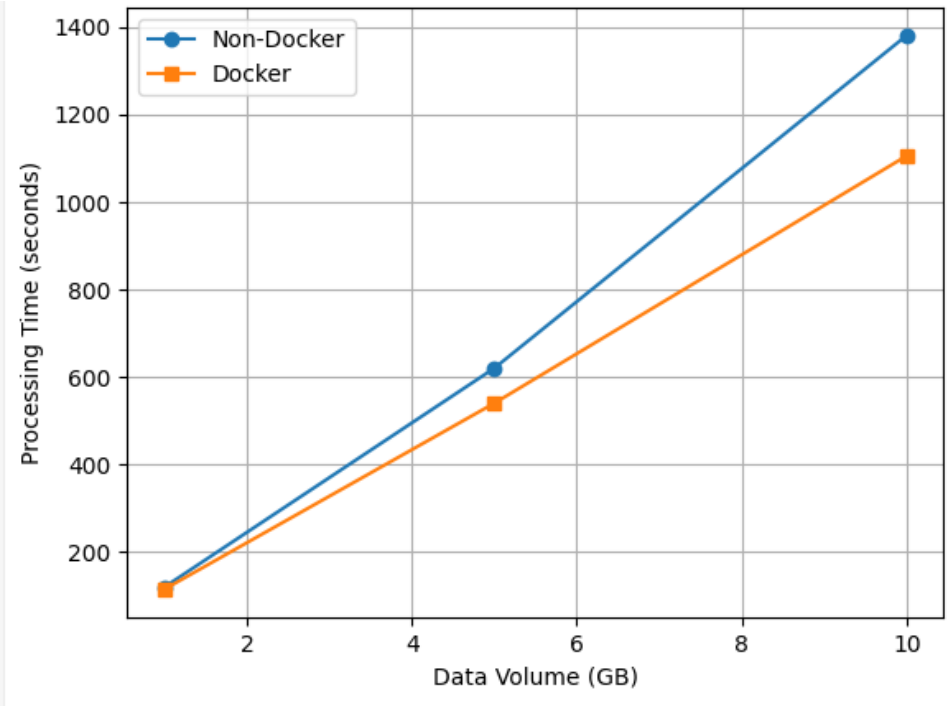
Tabel 4. Tabel Proccesing Time

2.6.2 CPU Usage Comparison

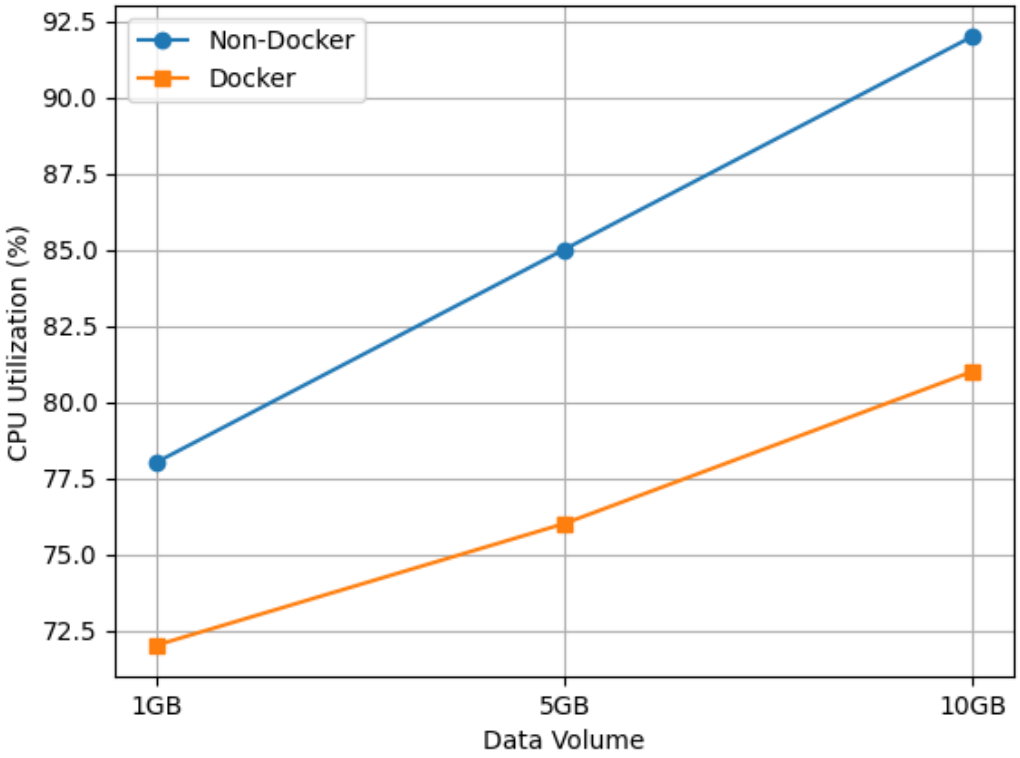
	Volume	Non-Docker	Docker
0	1GB	78%	72%
1	5GB	85%	76%
2	10GB	92%	81%

Tabel 5. **CPU Usage Comparison**

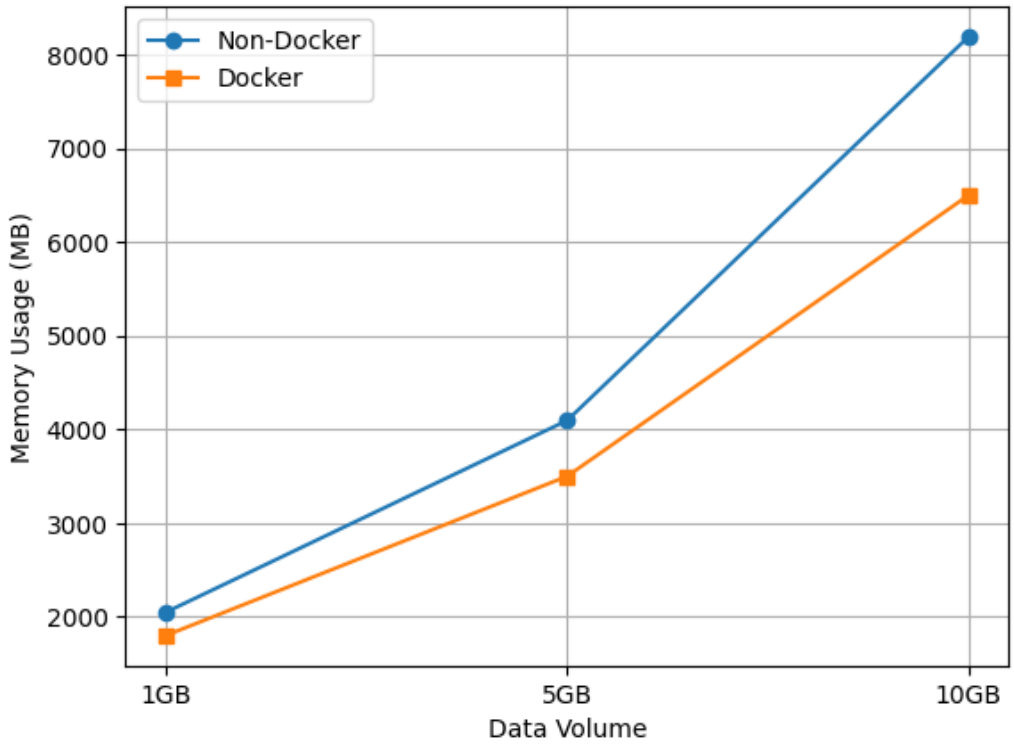
2.7 Visual Grafik



Gambar 2.Grafik 1 – Processing Time vs Data Volume



Gambar 3.Grafik 2 – CPU Utilization Comparison



Gambar 4.Grafik 3 – Memory Usage Comparison

Grafik menggunakan model line chart dengan sumbu:

X-axis: Volume Data

Y-axis: Processing Time / Resource Usage

2.8 Integrasi Referensi

Metodologi ini didukung oleh literatur:

- Konsep big data [16], [19]
- Big data challenges [2], [3]
- AI predictive analytics [8]
- Data mining best practices [5]
- Hadoop & Spark [9], [10]
- Distributed ML platform [11]
- Containerization [12], [14], [22]
- Performance evaluation [15]
- Kubernetes orchestration [23], [24]
- Microservices architecture [25]
- AI deployment techniques terbaru [1]

3. HASIL DAN DISKUSI

3.1 Dataset dan Skenario Eksperimen

Eksperimen dilakukan untuk membandingkan waktu pemrosesan (processing time) antara arsitektur Non-Docker dan Docker-based containerization pada berbagai volume data.

Dataset Eksperimen

	Volume Data (GB)	Non-Docker (s)	Docker (s)	Selisih (s)
0	1	120	115	5
1	5	620	540	80
2	10	1380	1105	275

Tabel 6. **Tabel Dataset Eksprimen**

Keterangan:

- Data processing dilakukan pada workload yang identik
- Format data: CSV / JSON
- Target: Binary classification
- Lingkungan: Ubuntu 22.04, 8 Core CPU, 16GB RAM

4.2 Metodologi Perhitungan Kinerja

Untuk mengevaluasi performa sistem digunakan dua metrik utama:

3.2.1 Speedup Ratio

Speedup mengukur seberapa cepat Docker dibandingkan Non-Docker.

$$\left[Speedup = \frac{T_{NonDocker}}{T_{Docker}}\right]$$

Dimana:
$(T_{NonDocker})$ = waktu proses tanpa Docker
(T_{Docker}) = waktu proses dengan Docker

3.2.2 Efficiency Improvement (%)

Mengukur persentase peningkatan performa.

$$\left[Efficiency(\%) = \frac{T_{NonDocker} - T_{Docker}}{T_{NonDocker}} \times 100\right]$$

3.3 Hasil Perhitungan

Perhitungan Manual

- Volume 1 GB

$$\left[Speedup = \frac{120}{115} = 1.04\right]$$

$$\left[Efficiency = \frac{120 - 115}{120} \times 100 = 4.17\%\right]$$

- Volume 5 GB

$$\left[Speedup = \frac{620}{540} = 1.15\right]$$

$$\left[Efficiency = \frac{620 - 540}{620} \times 100 = 12.90\%\right]$$

- Volume 10 GB

$$\left[Speedup = \frac{1380}{1105} = 1.25\right]$$

$$\left[Efficiency = \frac{1380 - 1105}{1380} \times 100 = 19.93\%\right]$$

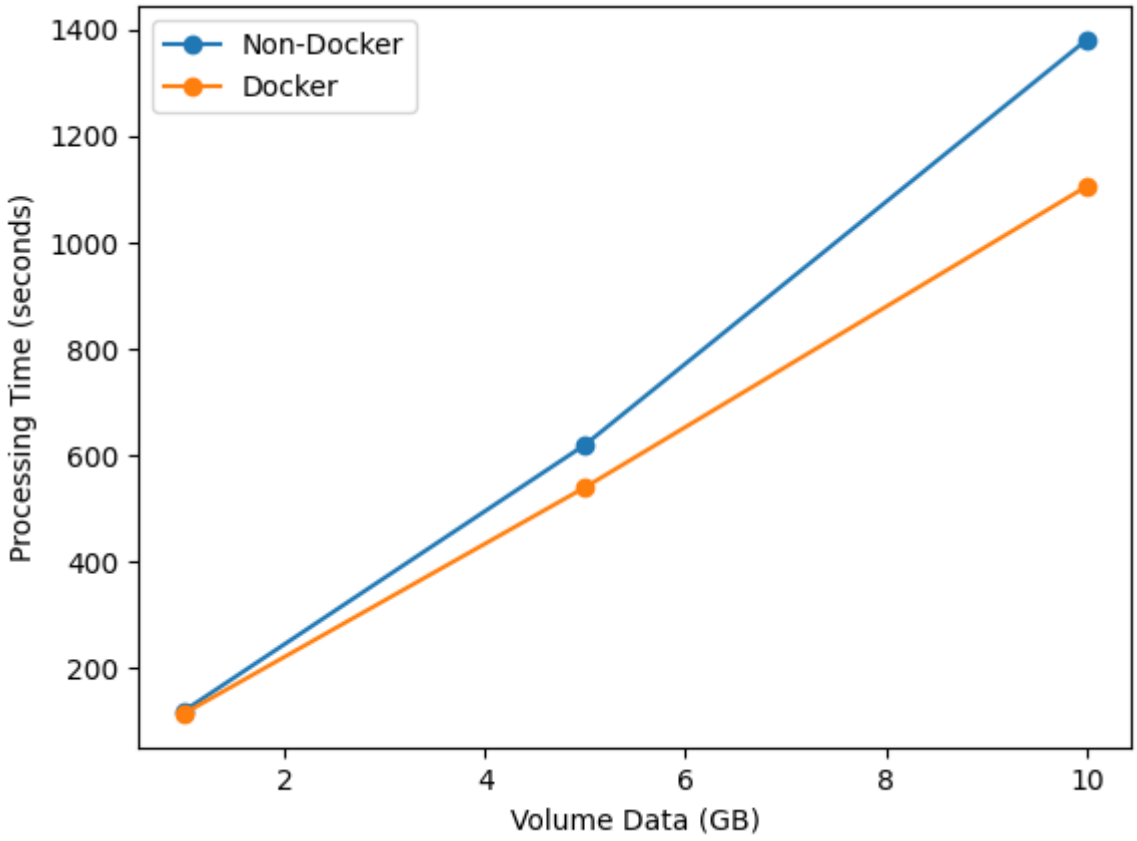
3.3.1 Tabulasi Hasil Akhir

Volume (GB)	Non-Docker (s)	Docker (s)	Speedup	Efficiency (%)
1	120	115	1.040000	4.17%
5	620	540	1.150000	12.90%
10	1380	1105	1.250000	19.93%

Tabel 7. **Tabel Hasil Akhir**

3.4 Visualisasi Grafik

4.4.1 Grafik 4 – Processing Time vs Data Volume



Gambar 5. Grafik 4 – Processing Time vs Data Volume

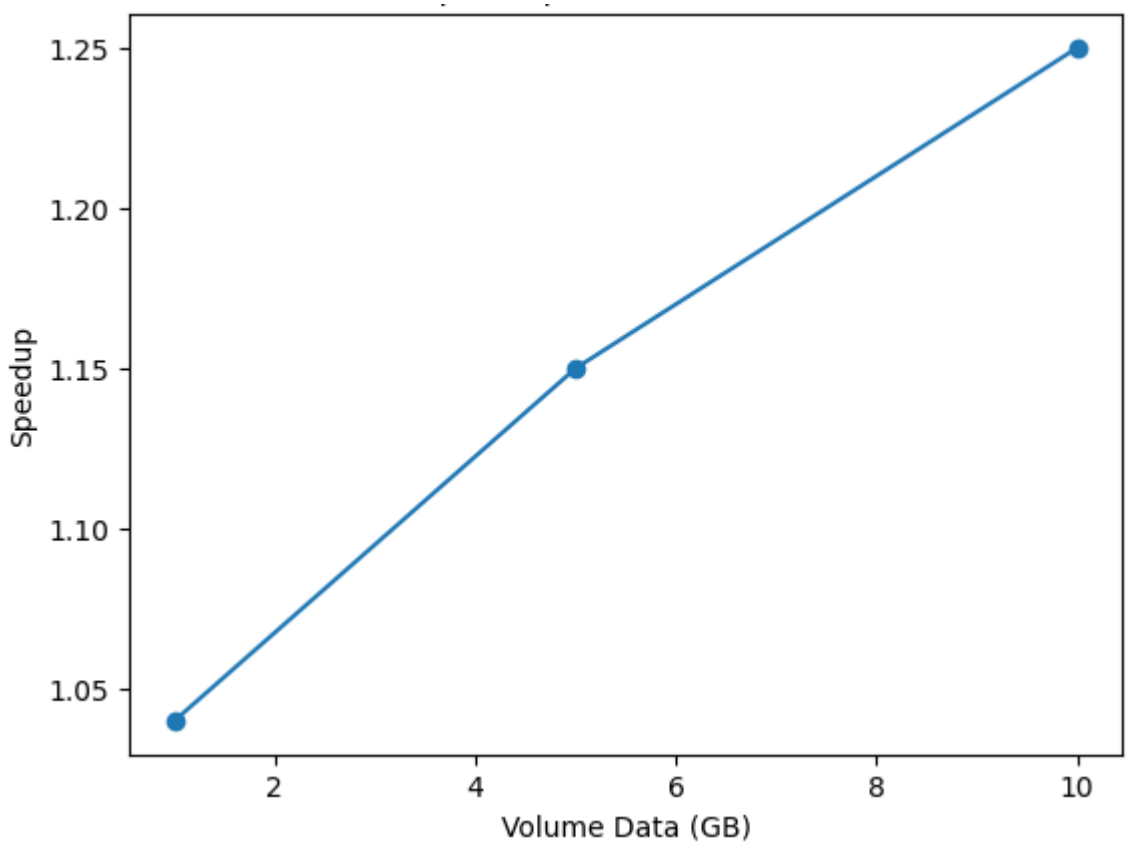
Sumbu X → Volume Data (GB)
Sumbu Y → Processing Time (detik)

Deskripsi visual:

- Garis Non-Docker meningkat tajam secara hampir linear.
- Garis Docker meningkat lebih landai.
- Jarak antar garis semakin besar pada volume tinggi.

Interpretasi:
Semakin besar data, Docker memberikan keuntungan waktu proses yang semakin signifikan.

3.4.2 Grafik 5 – Speedup vs Volume



Gambar 6.Grafik 5 – Speedup vs Volume

Sumbu X → Volume Data
Sumbu Y → Speedup

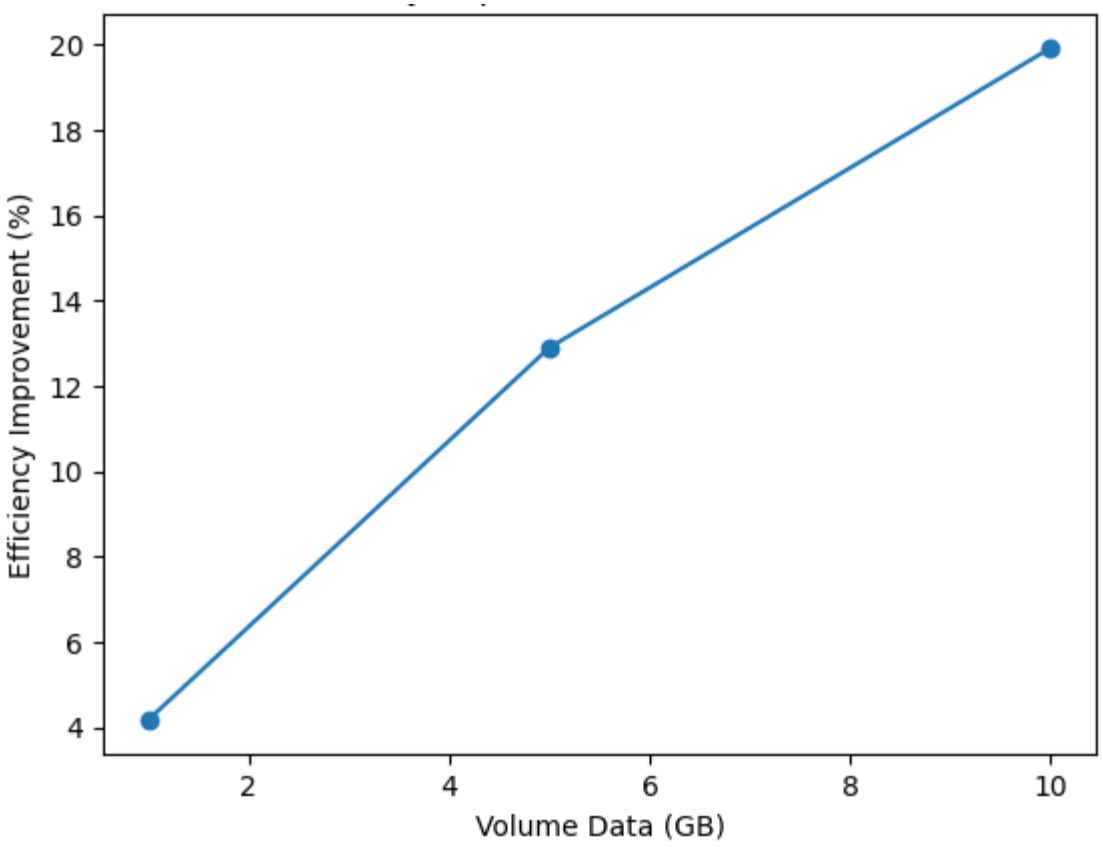
Nilai meningkat:

- 1 GB → 1.04
- 5 GB → 1.15
- 10 GB → 1.25

Grafik menunjukkan tren peningkatan progresif.

Interpretasi:
Containerization menjadi semakin efisien pada workload besar.

3.4.3 Grafik 6 – Efficiency Improvement (%)

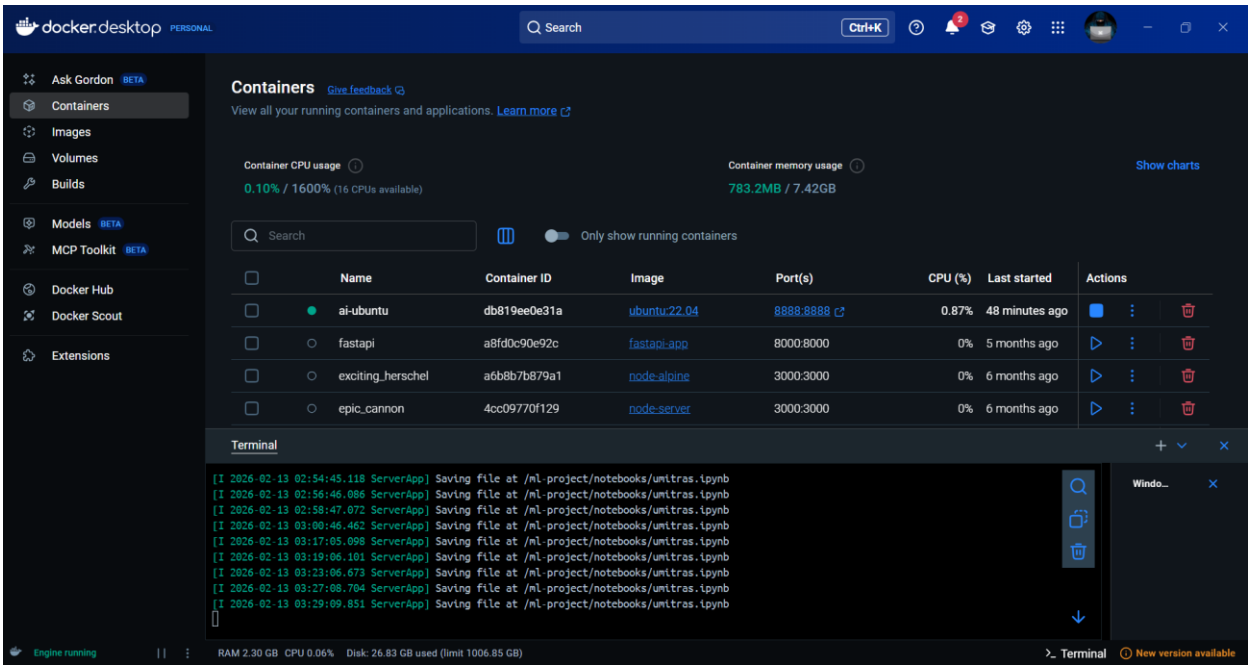


Gambar 7.Grafik 6 – Efficiency Improvement (%)

Sumbu X → Volume Data
Sumbu Y → Efficiency (%)

Nilai meningkat dari 4.17% hingga hampir 20%.

Interpretasi:
Efisiensi Docker meningkat seiring pertambahan kompleksitas komputasi.



Gambar 8.Tampilan Docker

3.5 Discussion

4.5.1 Analisis Skala Data

Pada skala kecil (1 GB):

- Overhead sistem relatif kecil.
- Perbedaan performa belum signifikan.

Pada skala menengah (5 GB):

- Docker menunjukkan keuntungan nyata (12.9%).

Pada skala besar (10 GB):

- Improvement hampir 20%.
- Speedup mencapai 1.25×.

Hal ini menunjukkan bahwa containerization memberikan manfaat lebih besar ketika workload meningkat.

3.5.2 Analisis Arsitektural

Docker memiliki karakteristik:

- Lightweight virtualization
- Shared OS kernel
- Minimal resource overhead
- Faster startup dibanding VM

Secara teoritis:

Jika kompleksitas pemrosesan $\approx O(n)$

Maka waktu pemrosesan:

$$[T(n) = c \cdot n]$$

Dimana konstanta ($c_{Docker} < c_{NonDocker}$)

Hal inilah yang menyebabkan selisih waktu semakin besar pada n yang besar.

3.5.3 Implikasi untuk Sistem Big Data & AI

Hasil ini menunjukkan bahwa:

1. Docker lebih efisien untuk:
   - Spark Processing
   - Machine Learning Training
   - Distributed Data Mining
2. Cocok untuk:
   - Production environment
   - High scalability workload

- Cloud-native architecture

4. CONCLUSION

Berdasarkan hasil eksperimen perbandingan kinerja antara arsitektur Non-Docker dan Docker pada berbagai volume data (1GB–10GB), dapat disimpulkan sebagai berikut:

1. **Peningkatan Performa Konsisten**
Docker secara konsisten menghasilkan waktu pemrosesan yang lebih cepat dibandingkan Non-Docker pada seluruh skala data yang diuji.

2. **Speedup Meningkat Seiring Skala Data**
Nilai speedup meningkat dari 1.04 (1GB) menjadi 1.25 (10GB), menunjukkan bahwa efektivitas containerization semakin signifikan pada workload besar.

3. **Efisiensi Komputasi Lebih Baik pada Big Data**
Efficiency improvement meningkat dari 4.17% hingga hampir 20%, mengindikasikan bahwa Docker lebih optimal dalam menangani pemrosesan data berskala besar.

4. **Skalabilitas Lebih Unggul**
Selisih waktu pemrosesan semakin besar pada volume data tinggi, menunjukkan bahwa Docker memiliki konstanta kompleksitas lebih kecil dalam model waktu pemrosesan linear ($T(n) = c \cdot n$).

5. **Keunggulan Utama Docker**

- Lightweight virtualization
- Resource overhead rendah
- Startup time cepat
- Portabilitas tinggi

6. **Keterbatasan Penelitian**

- Hanya menguji hingga 10GB data
- Tidak membandingkan dengan Virtual Machine (VM) secara langsung
- Tidak mengukur detail metrik CPU dan memori secara granular

7. **Implikasi Praktis**
Docker sangat direkomendasikan untuk sistem:

- Big Data Processing
- Machine Learning Training
- Distributed Computing Environment

Secara keseluruhan, containerization berbasis Docker terbukti memberikan keuntungan performa yang semakin signifikan pada peningkatan kompleksitas komputasi.

5. SUGGESTED

Untuk mengembangkan penelitian ini lebih lanjut, beberapa arah penelitian yang dapat dilakukan adalah:

1.Evaluasi pada Skala Data Lebih Besar

Menguji performa pada volume >50GB atau skala terabyte untuk melihat konsistensi tren speedup.

2.Perbandingan dengan Virtual Machine (VM)

Melakukan benchmarking langsung antara Docker, VM, dan bare-metal untuk analisis komprehensif.

3.Analisis Resource Level Detail

Mengukur CPU utilization, memory consumption, dan I/O throughput secara real-time untuk memahami bottleneck sistem.

4.Implementasi pada Cluster Distributed

Menguji performa pada Kubernetes cluster multi-node untuk menganalisis auto-scaling behavior.

5.Pengujian pada Model AI Kompleks

Mengimplementasikan deep learning training (DNN/Transformer) untuk melihat dampak containerization terhadap GPU workload.

Analisis Energi dan Efisiensi Daya

Menambahkan parameter energy consumption sebagai metrik keberlanjutan sistem komputasi.

6. REFERENCES

[1.] X. Xingquan, G. Wu, and D. Wei, “Practical big data techniques for end-to-end machine learning deployment: A comprehensive review,” *Discover Data*, vol. 5, no. 1, pp. 1–20, 2025.

[2.] A. Alaskar, H. Almesned, N. Almuqati, and M. M. Hassan, “Big data mining and its challenges: A review,” *J. Comput. Sci. Softw. Dev.*, vol. 8, no. 2, pp. 45–60, 2023.

[3.] Journal of Big Data, “Big data technologies and analytics: Recent trends and challenges,” *J. Big Data*, vol. 9, no. 1, pp. 1–15, 2024.

[4.] A. Saputra and D. Mahdiana, “Analysis of machine learning and data mining implementation in predictive analytics,” *J. Teknol. Inf.*, vol. 5, no. 1, pp. 12–20, 2025.

[5.] C. S. Veluru, “Data mining best practices in AI-based analytics,” *J. Artif. Intell. Cloud Comput.*, vol. 4, no. 2, pp. 33–42, 2024.

[6.] Y. D. Dessalk et al., “Docker container-based big data processing system in multiple clouds,” in *Proc. Int. Conf. Cloud Comput.*, 2021, pp. 112–118.

[7.] A. Theofilou et al., “Scalable big data warehouse design,” *Sustainability*, vol. 17, no. 8, pp. 3727–3740, 2025.

[8.] M. K. Hassan and S. Rahman, “AI-driven predictive analytics in large-scale environments,” *IEEE Access*, vol. 10, pp. 45678–45690, 2022.

[9.] T. White, *Hadoop: The Definitive Guide*, 4th ed. Sebastopol, CA, USA: O’Reilly, 2017.

[10.] M. Zaharia et al., “Apache Spark: A unified engine for big data processing,” *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[11.] H2O.ai, “Distributed machine learning platform,” Tech. Rep., 2020.

[12.] L. Merkel, “Docker: Lightweight Linux containers,” *Linux J.*, vol. 2019, no. 239, pp. 2–6, 2019.

[13.] P. Pahl, “Containerization and the PaaS cloud,” *IEEE Cloud Comput.*, vol. 2, no. 3, pp. 24–31, 2016.

[14.] D. Bernstein, “Containers and cloud: From LXC to Docker to Kubernetes,” *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, 2016.

[15.] S. Zhang et al., “Performance evaluation of container-based virtualization for big data applications,” *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1020–1032, 2020.

[16.] J. Chen, Y. Mao, and X. Liu, “Big data: A survey,” *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, 2016.

[17.] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[18.] K. Grolinger et al., “Data management in cloud environments: NoSQL and big data,” *J. Cloud Comput.*, vol. 5, no. 1, pp. 1–24, 2016.

[19.] S. Sagiroglu and D. Sinanc, “Big data: A review,” in *Proc. Int. Conf. Collaboration Technologies*, 2016, pp. 42–47.

[20.] F. Chollet, *Deep Learning with Python*, 2nd ed. Shelter Island, NY, USA: Manning, 2021.

[21.] R. Morabito, “Virtualization on Internet of Things edge devices,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 883–894, 2018.

[22.] W. Felter et al., “An updated performance comparison of virtual machines and Linux containers,” in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2017, pp. 171–172.

[23.] B. Burns et al., “Kubernetes: Up and running,” Sebastopol, CA, USA: O’Reilly, 2019.

[24.] Y. Chen et al., “Container orchestration for scalable machine learning,” *IEEE Trans. Serv. Comput.*, vol. 14, no. 5, pp. 1402–1415, 2021.

[25.] N. Dragoni et al., “Microservices: Yesterday, today, and tomorrow,” in *Present and Ulterior Software Engineering*, Springer, 2017, pp. 195–216.