The Indonesian Journal of Computer Science
www.ijcs.net
Volume 13, Issue 4, August 2024
https://doi.org/10.33022/ijcs.v13i4.4178

Perbandingan Penggunaan Daya Listrik Antara *Baremetal* Dengan *Docker Container* Pada Raspberry Pi 4B

Hasvid Dwi Putranto[1], Muhammad Koprawi[2]
hasvid@students.amikom.ac.id, koprawi@amikom.ac.id
[12] Fakultas Ilmu Komputer, Program Studi Teknik Komputer, Universitas Amikom Yogyakarta

Informasi Artikel

Diterima : 23 Jun 2024
Direview : 2 Jul 2024
Disetujui : 25 Jul 2024

Kata Kunci

Docker, HPLinpack, IOPS, Edge Server, Flops

Abstrak

Pada perkembangan jaman sekarang, banyak layanan yang mulai memakai *Container base service* dimana lebih mudah melakukan *development* dan dokumentasi penerapan nya juga banyak. Tetapi ada juga yang masih memakai *Baremetal* dimana memasang layanan tersebut dipasang langsung di *Operating System* (OS) dan beberapa orang terbiasa memakai *Baremetal* dalam hal ini penerapan di *Edge Server* atau *Internet of Things* (IoT) seperti pada Raspberry Pi, penggunaan listrik merupakan limitasi dalam membuat sistem yang efisien sehingga dalam penelitian ini, akan membandingkan penggunaan listrik dalam *Input/output operations per second* (IOPS) dan *Floating point operations per second* (Flops). Dan dalam hasil penelitian menunjukkan bahwa penggunaan listrik *Baremetal* lebih kecil dari pada *Docker Container* dimana perbedaan nya 1,06 *GigaFlops/W* pada *HPLinpack* dan pada *IOPS/Watt*, *Baremetal* jauh lebih unggul dari pada *Docker Container* dimana pada pada *IOPS read/Watt* unggul 3246,12 dan pada IOPS write/Watt unggul 1857,51 hal itu disebabkan karena *Docker Container* menggunakan beberapa *service* untuk menjalankan *Container*-nya sehingga mengkonsumsi daya yang lebih banyak.

Keywords

Docker, HPLinpack, IOPS, Edge Server, Flops

Abstract

In today's development, many services are starting to use Container base services where it is easier to do development and there is also a lot of implementation documentation. But there are also those who still use Baremetal where installing the service is installed directly on the Operating System (OS) and some people are used to using Baremetal in this case the application on the Edge Server or Internet of Things (IoT) such as on the Raspberry Pi, electricity usage is a limitation in making an efficient system so that in this study, will compare electricity usage in Input/output operations per second (IOPS) and Floating point operations per second (Flops). And the results show that Baremetal's electricity usage is smaller than that of Docker Container where the difference is 1.06 GigaFlops/Watt on HPLinpack and on IOPS/Watt, Baremetal is far superior to Docker Container where in IOPS read/Watt superior to 3246.12 and in IOPS write/Watt superior to 1857.51 it is because Docker Container uses several services to run its Container so that it consumes more power.

A. Pendahuluan

Di era digital ini, banyak perusahaan besar mulai beralih menggunakan layanan *container* dan *serverless*. Teknologi ini secara perlahan menggantikan penggunaan *Baremetal*, di mana aplikasi dipasang langsung ke OS [1]. Alasan utama peralihan ini adalah karena *container* dan *serverless* menawarkan beberapa keunggulan, seperti lebih ringan dan mudah di-*scale* [2].

Penerapan *container* dan *serverless* sudah diterapkan di beberapa *website* dan *Cloud Platform* mulai menawarkan layanan *Function as a Service* (FaaS) [3], [4]. FaaS memanfaatkan teknologi *serverless* dan menyediakan dokumentasi yang menunjang pengguna untuk melakukan pengembangan dengan mudah di layanan tersebut.

Di sisi lain, penerapan aplikasi langsung ke *Operating System* (OS) memang mudah dalam hal konfigurasi. Banyak langkah-langkah penerapannya tersedia di internet dengan penjelasan yang detail, sehingga banyak pengguna yang masih menggunakannya. Namun, masih ada beberapa orang yang belum mencoba *container*, sehingga ada beberapa penerapan layanan yang akan memakan resource yang besar, seperti listrik, CPU, dan RAM, sehingga banyak yang terbuang sia-sia[5].

Penelitian ini berfokus pada penggunaan *Docker Container* [6], [7] dan menggunakan OS ubuntu server LTS. Dalam penelitian ini, dilakukan perhitungan mulai dari *CPU usage*, RAM, dan *benchmark watt/flops*. Hal ini bertujuan untuk membandingkan kinerja dan efisiensi penggunaan *resource* antara aplikasi yang dijalankan langsung ke OS dan aplikasi yang dijalankan dalam *container*.

Penelitian ini juga akan meneliti perbedaan penggunaan *resource* antara aplikasi *container* dan aplikasi *serverless*. Di beberapa *Home Server*, aplikasi masih dipasang langsung ke OS karena dianggap mudah digunakan. Namun, banyak yang beranggapan bahwa penggunaan *resource* aplikasi yang dijalankan langsung ke OS akan sama dengan *container* dan menghasilkan performa yang sama. Penelitan ini akan mencoba untuk mencari perbedaan tersebut [3], [8], [9].

Berdasarkan penelitian sebelumnya, yang dilakukan oleh Pablo Josue Rojas Yepes dan Carlos Jaime Barrios Hernandez dengan judul "Impact of Containerization on Low-Cost Post Moore Computing Architectures" (2022)[10], dimana mereka melakukan perbandingan dengan *docker, singularity*, dan *native*. Mereka menyimpulkan dampak pada *I* dan performa sangat kecil dan tidak berpengaruh pada metode penerapan tetapi tergantung pada *resource* yang ada.

Dan berdasarkan penelitian Xavier Merino dengan judul "The Cost of Virtualizing Time in Linux Containers" (2022)[11], mereka melakukan penelitian waktu yang dibutuhkan untuk Virtualisasi dalam kontainer menyimpulkan bahwa terdapat pengaruh kurang dari 4% dengan *system call* yang meningkat sebesar 2% dimana dibegaruhi oleh waktu dalam *container* yang migrasi.

Kemudian penelitian yang dilakukan oleh Shahidullah Kaiser dalam judul "Benchmarking Container Technologies on ARM-Based Edge Devices" (2023)[12], melakukan bechmark pada *container* yang memakai *architecture Advanced RISC Machines* (ARM) dan melakukan testing untuk *Artificial Intellegent* dan menemukan bahwa dengan menggunakan *Singularity* bekerja dengan baik dalam *data science workload*, sedangkan tengnologi *Docker container* memiliki *CPU Usage*

yang kecil pada saat *task facial detection* dan menentukan bahwa setiap teknologi memiliki keterbatasan pada masing masing bidang nya.

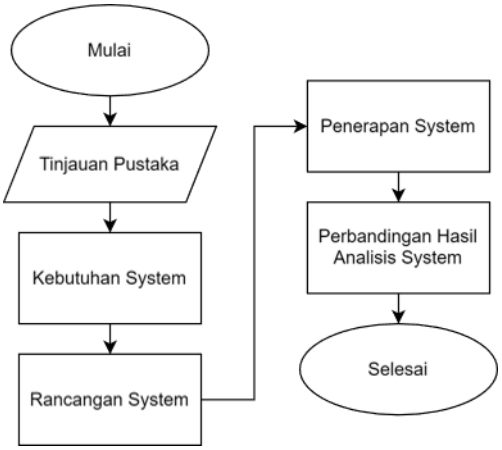
Dan pada penelitian yang dilakukan oleh Tom Goethals dengan judul "A Functional and Performance Benchmark of Lightweight Virtualization Platforms for Edge Computing" (2022)[2]. Dimana melakukan perbandingan dengan *OSv, Firecracker, Docker,* dan *gVisor*. Dalam melakukan perbandingan tersebut dia menemukan bahwa setiap teknologi tersebut memiliki keunggulan dan kekurangan seperti *docker* miliki performa yang bagus dan scaling pada *core* yang bagus tetapi memiliki *latency* yang tidak stabil dan kerentanan keamanan.

Sedangkan pada penelitian yang dilakukan oleh Michael Sollfrank dengan judul "Evaluating Docker for Lightweight Virtualization of Distributed and Time-Sensitive Applications in Industrial Automation" (2021)[13], menyimpulkan bahwa *docker* sudah memiliki ketentuan untuk melakukan *automation* tetapi memiliki masalah pada waktu penjalanan aplikasi dimana menambah *delay* pada *node*-nya.

B. Metode Penelitian

a. Tahapan penelitian

Tahapan penelitian merupakan suatu rencana penelitian untuk menerapkan sistem yang dibuat oleh peneliti, dalam penelitian ini akan melakukan beberapa tahapan seperti pada Gambar 1 Diagram Penelitian.



Gambar 1. Diagram Penelitian

1. Tinjauan Pustaka

Dalam bagian ini, peneliti akan mempelajari dan memahami penelitian sebelum yang sudah ada yang berkaitan dengan tema penelitian, dalam penelitian ini berfokus pada perbandingan penggunaan listrik antara *Docker Container* dengan *Native* atau *Baremetal* pada Raspberry Pi 4B.

2. Kebutuhan Sistem

Bagian ini merupakan proses untuk memahami perangkat apa saja yang akan dibutuhkan mulai dari *hardware* maupun *software*, dalam kebutuhan ini dapat dilihat pada Tabel 1 *Software* Raspberry Pi 4B di bawah untuk melihat kebutuhan *software* yang dipakai pada Raspberry Pi 4B.

Tabel 1. *Software* Raspberry Pi 4B

No.	Software	Detail	Version
1	Raspbian Bookworm	*Operating System* yang akan dijalankan di Raspberry Pi 4B	Kernel 6.1.0-rpi7-rpi-v8
2	Docker Client	Aplikasi untuk menjalankan suatu *command*	26.1.1
3	Docker Engine	Sistem yang mengatur *API* dan fungsi lain *docker*	26.1.1
4	Containerd	Sistem yang mangatur *life cycle container*	1.6.31
5	runc	Sistem yang mengatur *envirotment container*	
6	Docker-init	Sistem yang meng-*initialisasi project* untuk memberikan *file* yang dibutuhkan dalam menjalankan *container*	0.19.0
7	Samba	Aplikasi *file network* untuk melakukan pengetesan *iops*	4.17.12-Debian
4	HPLinpack	Aplikasi untuk melakukan *test GFlops*	2.3-BLIS

Sedangkan untuk *software* yang digunakan di PC *Benchmark* bisa melihat pada Tabel 2 *Software* PC *Benchmark* di bawah.

Tabel 2. *Software* PC *Benchmark*

No.	Software	Detail	Version
1	Windows CMD	Digunakan untuk mengakses *Server Ubuntu* dan Raspberry melalui *SSH Client*	10.0.19045.4291
2	Cystall Disk Mark	Melakukan *benchmark* pada *Server Message Block* (SMB) dan menampilkan hasil performanya	8.0.4 x64

Kemudian dalam melakukan penelitian ini, menggunakan *hardware* dan *software* yang bisa dilihat pada Tabel 3 *Hardware* di bawah.

Tabel 3. Hardware

No.	Kategori	Spesifikasi	Version\Firmware\Name\Driver
1	OS	Windows 10	22H2 Build 19045.4291
2	RAM	32 GB DDR4 @ 2666MHz	G.Skill F4-2666C19-16GVR
3	CPU	AMD Ryzen 9 3900X @ 4.1 GHz	-
4	NIC	Marvell AQtion 10Gbit	Driver 3.1.7.0
5	Power Monitoring	SONOFF POWR320D	Firmware 1.0.7
6	Powerwall	Baseus Wall Charger 100W	CCGAN100CE
7	Cable LAN	CAT-6e	-

8	Raspberry Storage	Sandisk Ultra Flair	16GB
9	Raspberry Cooling	*Aluminium heatsink case with single Fan*	-

3. Rancangan Sistem

   Dalam membuat rancangan Sistem ini peneliti membuat sebuah rancangan infrastruktur yang digunakan untuk melakukan sebuah penelitian dimana akan dibuat melalui gambaran topologi supaya dapat dengan mudah dipahami.
4. Penerapan Sistem

   Dalam melakukan penerapan ini sesuai dengan rancangan dan kebutuhan Sistem yang akan dipakai dalam melakukan penelitian.
5. Perbandingan Hasil Analisis Sistem

   Membandingkan hasil yang telah dikumpulkan melalui *monitoring* dan observasi yang telah dilakukan pada Sistem.

b. Metode Pengumpulan Data

Dalam melakukan pengumpulan data dimulai dari observasi Sistem yang telah dikumpulkan dan memasukan-nya kedalam *Spreadsheet* supaya dapat dengan mudah divisualisasikan kemudian melakukan observasi pada *power meter* untuk mengetahui seberapa banyak listrik yang dipakai.

c. *Docker Container*

Merupakan suatu *software* dimana untuk dimudahkan dalam membuat, menerapkan, dan mengeksekusi suatu aplikasi mengunakan teknologi *container*. Dari kelebihan tersebut *Docker* merupakan salah satu alat yang paling banyak digunakan untuk DevOPS untuk melakukan suatu *automasi* dalam CI/CD. *Docker* juga bisa menjadi *alternative* penggunaan *Virtual machine* karena sama-sama menggunakan *virtual envirotment*.

d. Pengunaan Daya Listrik

Dalam menghitung penggunaan daya listrik, penelitian ini menggunakan parameter sebagai berikut:

1. *Flops/Watt*

   Floating point operations per second (*Flops*) merupakan salah satu pengukuran *computing* dalam mengitung performa *computing*. Dalam hal ini, membandingkan apakah *docker* memperngaruhi *Flops* dan penggunaan listrik tersebut. Dalam penelitian ini menggunakan *formula* di bawah [14], [15].

$$\text{Gflops/W} = \frac{\text{Gflops}}{\text{Watt}}$$

Rumus 1. Gflops/Watt

2. *Iops/Watt*

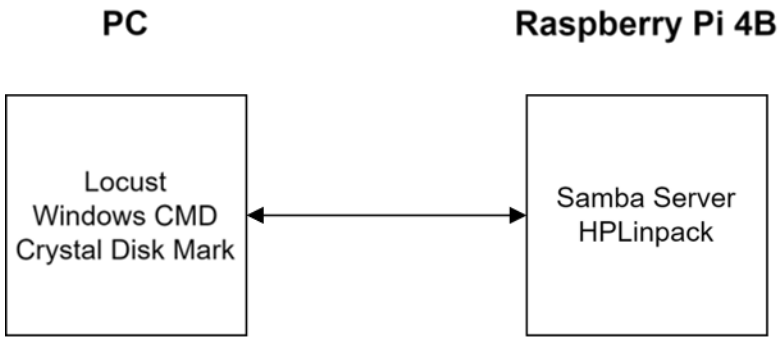
Input/output operations per second (*iops*) merupakan salah satu alat ukur *input/output* yang digunakan untuk melakukan *benchmark* pada suatu *storage*. Dalam melakukan *test* ini, menggunakan rumus di bawah[16], [17] dimana hasil *iops* dibagi dengan *peak watt* yang di pakai.

$$\text{iops/W} = \frac{\text{iops}}{\text{Watt}}$$

Rumus 2. Iops/Watt

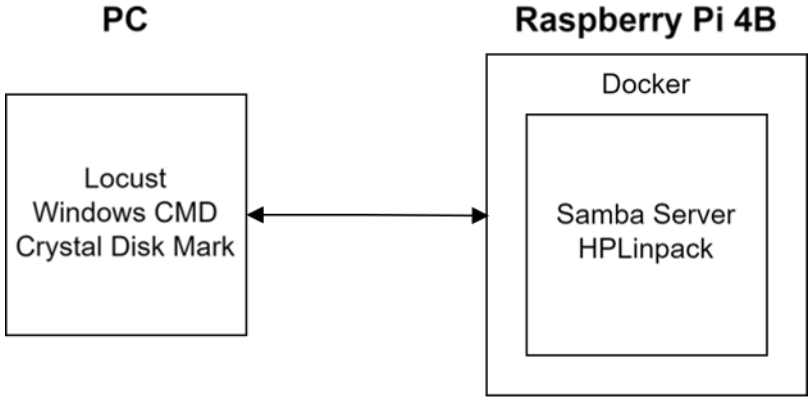
e. Rancangan Sistem

Dalam melakukan rancangan sistem ini, penelitian ini berusaha melakukan komparasi secara *apple to apple* yaitu berusaha supaya *hardware* tetap sama dan menghasilkan rancangan desain nya seperti Gambar 2 Rancangan sistem *Baremetal* di bawah dimana tidak menggunakan *Docker Container*



Gambar 2. Rancangan Sistem *Baremetal*

Sedangkan untuk rancangan sistem yang menggunakan *Docker Container* bisa dilihat pada Gambar 3 Rancangan Sistem *Docker Container* di bawah



Gambar 3. Rancangan Sistem *Docker Container*

f. Skenario pengujian

Penelitian ini akan mengadakan dua pengujian. Terlihat pada Gambar 2 dimana pengujian dilakukan pada *baremetal* atau langsung pada OS nya dan Gambar 3 dimana melakukan pengujian pada layanan yang telah dipasang di *Docker Container* dan bukan pada baremetal nya.

Kemudian setiap *testing* penelitian akan menggunakan parameter sebagai beriku:

1. *Crystall Disk Mark*.

   Untuk parameter yang dipakai *Crystall Disk Mark* sesuai pada Table 4 Parameter *Crystall Disk Mark* di bawah.

Tabel 3. Parameter Crystall Disk Mark

Type	Block Size	Queues	Threads
SEQ	1MiB	8	1
SEQ	128KiB	32	1
RND	4KiB	32	16
RND	4KiB	1	1

2. *HPLinpack*

   Untuk configurasi HPL.dat nya, menggunakan *config* seperti pada Gambar 4 *Config* HPL.dat di bawah.

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
20352         Ns
1            # of NBs
192           NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
1            Ps
1            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
1            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
##### This line (no. 32) is ignored (it serves as a separator). ######
0                               Number of additional problem sizes for PTRANS
1200 10000 30000                values of N
0                               number of additional blocking sizes for PTRANS
40 9 8 13 13 20 16 32 64        values of NB
```

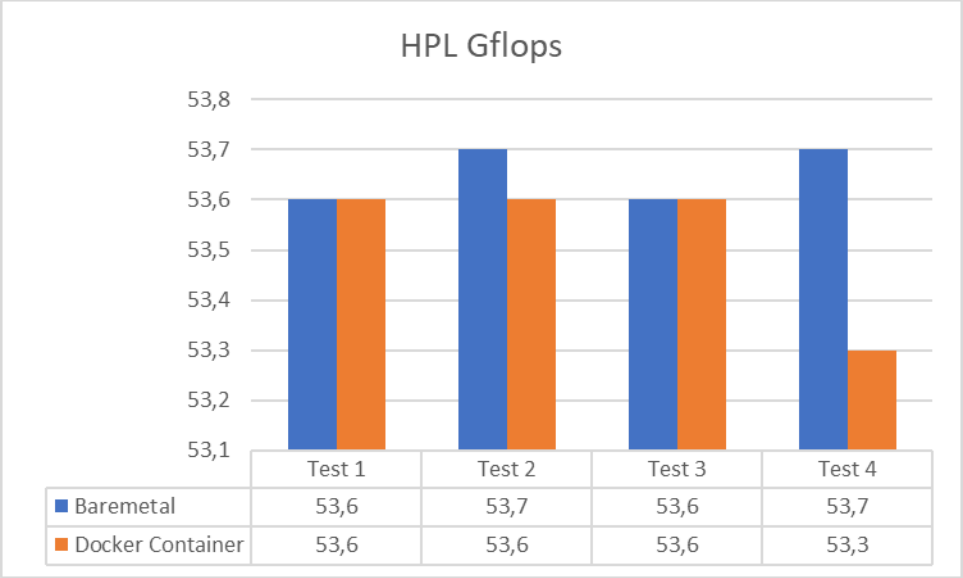
Gambar 4. *Config* HPL.dat

C. Hasil dan Pembahasan

Setelah melakukan persiapan dan penerapan sistem, kemudian melakukan pengujian dan mengamati penelitian, menghasilkan nilai sebagai berikut.

a. *HPLinpack*

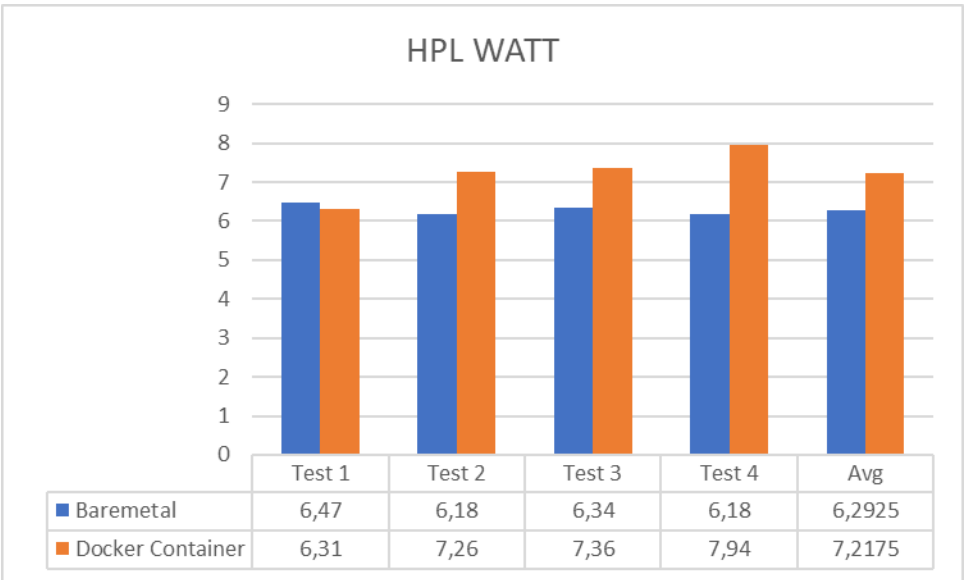
Pada aplikasi HPLinpack, penelitian ini melakukan *benchmark* sebanyak 4 kali dimana menghasilkan nilai *Gflops* dimana ditampilkan pada Gambar 5 *Benchmark HPLinpack* di bawah.



	Test 1	Test 2	Test 3	Test 4
Baremetal	53,6	53,7	53,6	53,7
Docker Container	53,6	53,6	53,6	53,3

Gambar 5. *Benchmark HPLinpack*

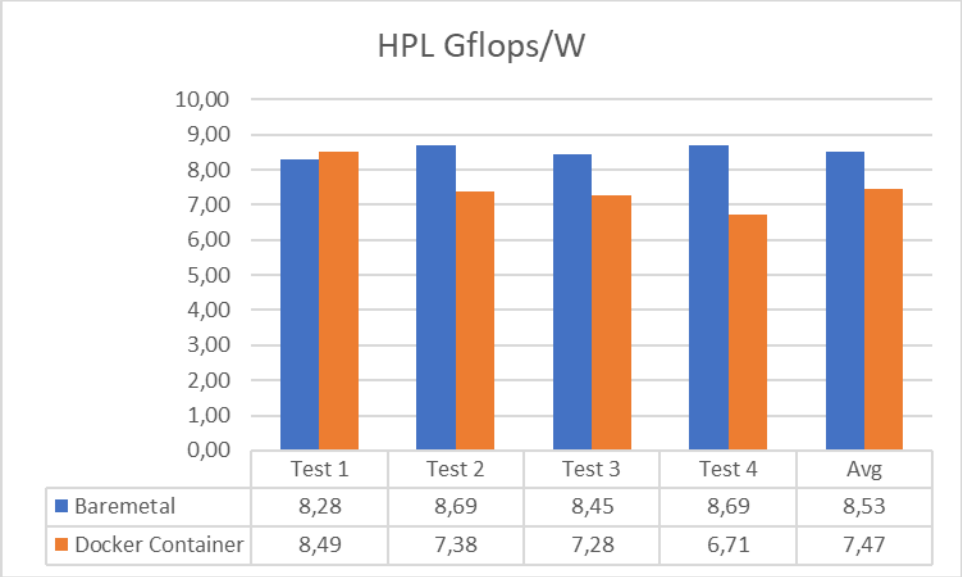
Pada hasil *benchmark* tersebut, perbedaan nya tidak begitu banyak, sedangkan dalam perbedaan pengunaan *peak watt* nya, dimana rata-rata penggunaan *watt Docker Container* lebih tinggi dari pada *baremetal* yang seperti ditampilkan pada Gambar 6 Penggunaan *Watt HPL* di bawah.



	Test 1	Test 2	Test 3	Test 4	Avg
Baremetal	6,47	6,18	6,34	6,18	6,2925
Docker Container	6,31	7,26	7,36	7,94	7,2175

Gambar 6. Penggunaan *Watt HPL*

Dan jika dibandingkan dengan Gflops/W nya *baremetal* lebih unggul dari pada *Docker Container* diamana rata-rata nya lebih tinggi dari pada *Docker Container* yang ditunjukan pada Gambar 7 Hasil *Gflops/Watt HPL* di bawah.



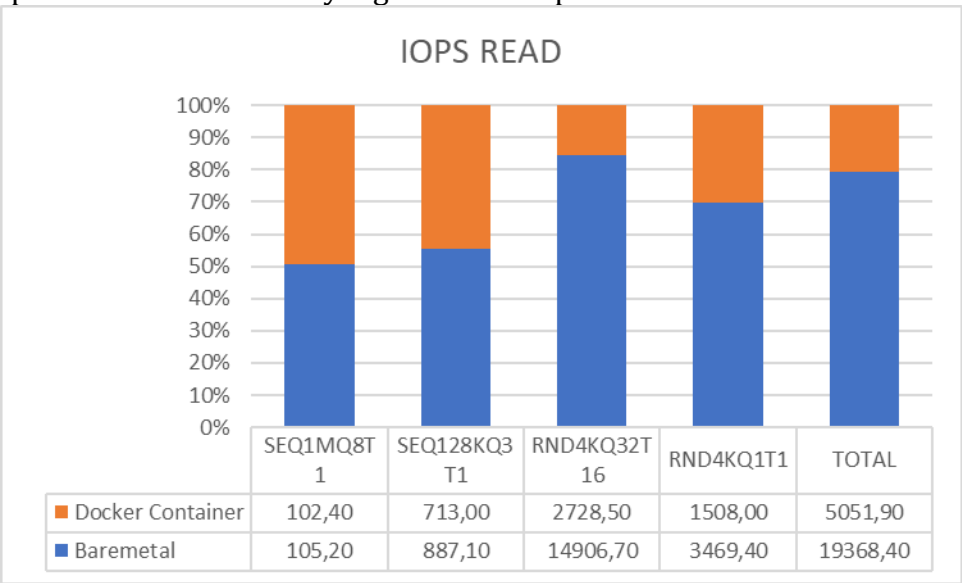
Gambar 7. Hasil *Gflops/Watt HPL*

b. Crystall Disk mark IOPS

Pada pengujian ini kita membandingkan *IOPS Read*, *IOPS Write*, dan begitu pula penggunaan *peak watt* nya, yang jelaskan sebagai berikut:

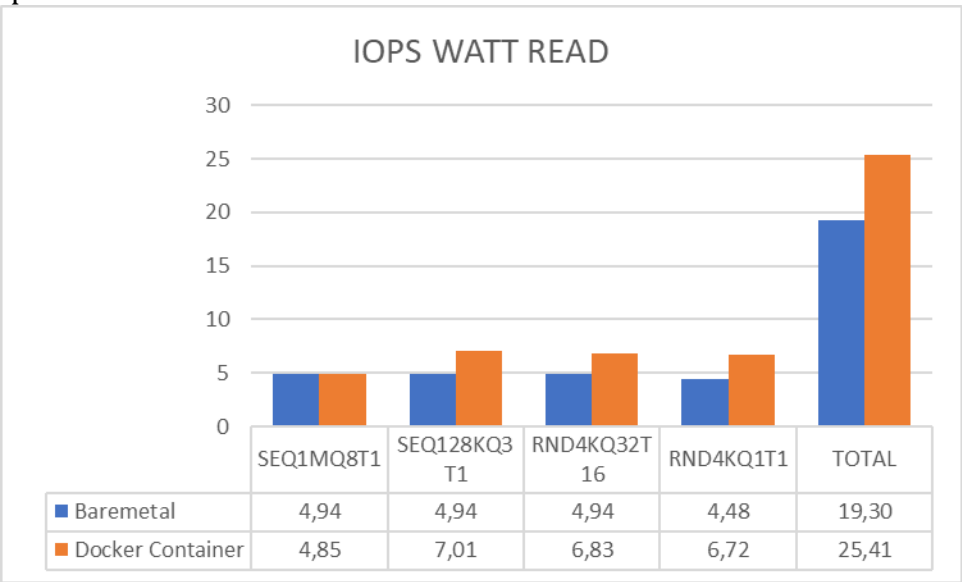
1. *IOPS Read*

Dalam IOPS read ini baremetal memiliki *IOPS* yang lebih tinggi dari pada docker container yang bisa dilihat pada Gambar 8 *IOPS read* di bawah.



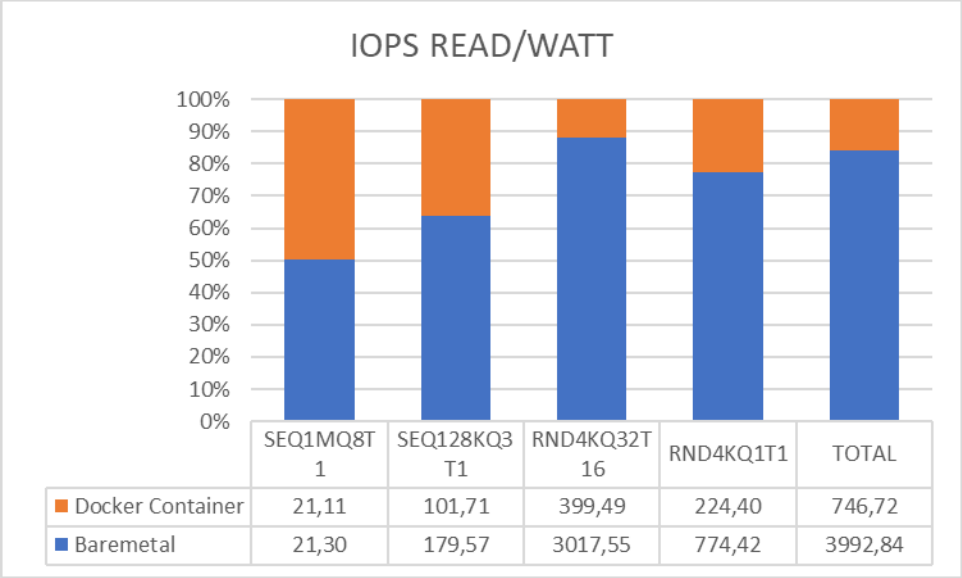
Gambar 8. *IOPS Read*

Dan dalam penggunaan *peak watt* nya *Docker Container* membutuhkan *watt* yang lebih besar dari pada *baremetal* yang bisa dilihat pada Gambar 9 *IOPS Watt Read* di bawah.



Gambar 9. *IOPS Watt Read*

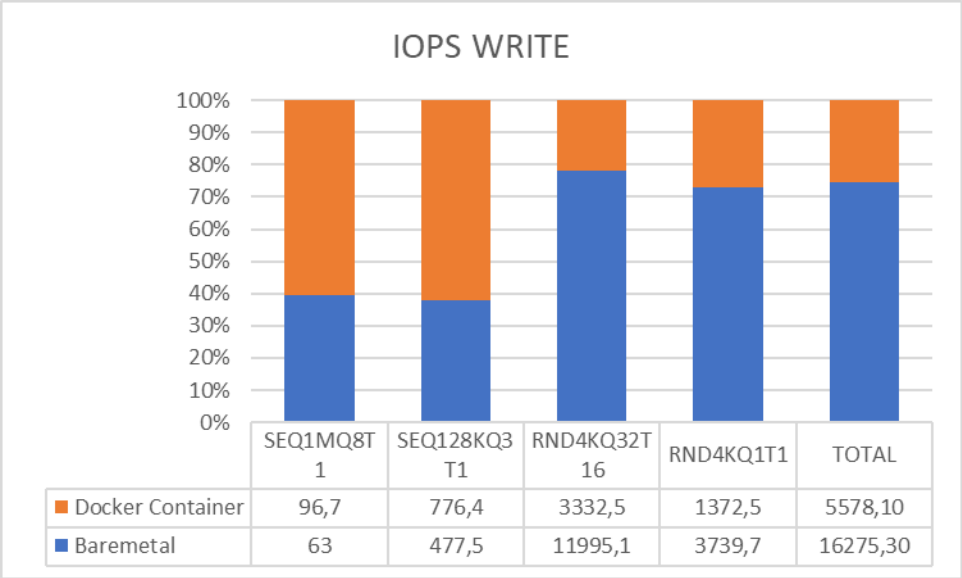
Sehingga hasil dari *IOPS*/W-nya, *baremetal* lebih unggul dari pada *Docker Container* yang dibisa pada Gambar 10 *IOPS*/*WATT Read* di bawah.



Gambar 10. *IOPS*/*Watt Read*

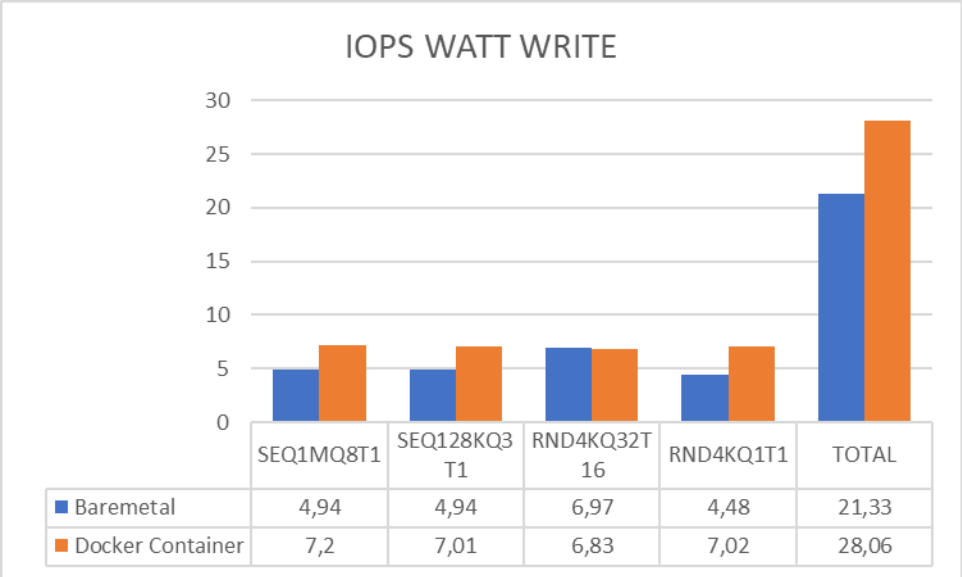
2. *IOPS Write*

Pada bagian *sequential 1M* dan *128K*, *Docker Container* lebih unggul dari pada *baremetal* sedangkan pada bagian *random 4K baremetal* lebih unggul yang bisa dilihat dari Gambar 11 *IOPS Write* di bawah.



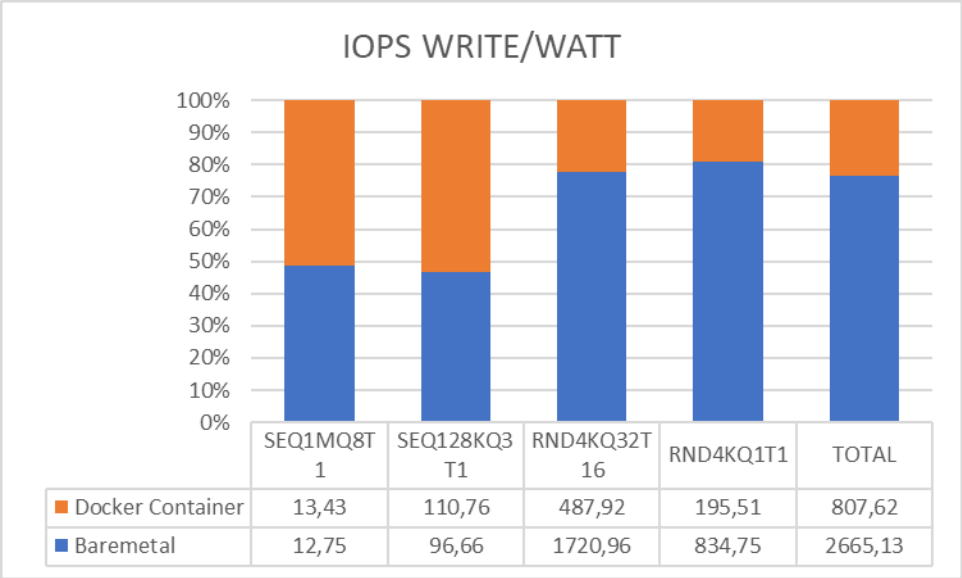
Gambar 11. *IOPS Write*

Sedangkan pada *peak watt*-nya *baremetal* menggunakan listrik sangat kecil dibandingkan dengan *Docker Container* yang bisa dilihat dari Gambar 12 *IOPS Watt Write* di bawah.



Gambar 12. *IOPS Watt Write*

Sedangkan pada *iops write*/*watt* dimana pada bagian *sequence*, perbedaan nya tidak terlalu banyak sedangkan pada bagian *random* perbedaan nya sangat signifikan yang ditunjukan pada Gambar 13 *IOPS Write*/*Watt* di bawah.



	SEQ1MQ8T1	SEQ128KQ3T1	RND4KQ32T16	RND4KQ1T1	TOTAL
Docker Container	13,43	110,76	487,92	195,51	807,62
Baremetal	12,75	96,66	1720,96	834,75	2665,13

Gambar 13. *IOPS Write/Watt*

D. Simpulan

Melalui penggujian tersebut, baremetal lebih unggul dalam performa dan peak watt nya terutama pada *IOPS* dimana *IOPS write/watt Docker Container* dalam RND4KQ32T mendapatkan nilai lebih kecil dari pada *Baremetal* tetapi dalam melakukan *test Gflops/Watt* dan *Gflops* nilai *Baremetal* dan *Docker Container* mendapat nilai yang hampir sama karena limitasi *hardware* dalam Raspberry Pi 4B.

Dan dalam hasil penelitian menunjukkan bahwa penggunaan listrik *Baremetal* lebih kecil dari pada *Docker Container* dimana perbedaan nya 1,06 *GigaFlops*/W pada *HPLinpack* dan pada *IOPS/Watt*, *Baremetal* jauh lebih unggul dari pada *Docker Container* dimana pada pada *IOPS read/Watt* unggul 3246,12 dan pada *IOPS write/Watt* unggul 1857,51.

Dalam hasil penegetesan tersebut *Docker Container* memiliki nilai yang cukup rendah dikarenakan harus memanggil beberapa *API* nya untuk mengakses *kernel* sehingga memiliki performa yang buruk sedangkan *Baremetal*, aplikasi yang dijalakan itu langsung mengakses *kernel* sehingga dapat mendapatkan performa yang lebih bagus.

E. Referensi

[1] X. Lyu, "Balancing Three Important Goals for Runtimes - Isolation, High Performance, and Resource Efficiency," in *Proceedings - 2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion, ACSOS-C 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 60–62. doi: 10.1109/ACSOSC56246.2022.00031.

[2] T. Goethals, M. Sebrechts, M. Al-Naday, B. Volckaert, and F. De Turck, "A Functional and Performance Benchmark of Lightweight Virtualization Platforms for Edge Computing," in *Proceedings - IEEE International*

Conference on Edge Computing, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 60–68. doi: 10.1109/EDGE55608.2022.00020.

[3] A. Tzenetopoulos *et al.*, "FaaS and Curious: Performance Implications of Serverless Functions on Edge Computing Platforms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 428–438. doi: 10.1007/978-3-030-90539-2_29.

[4] T. Pfandzelter and D. Bermbach, "TinyFaaS: A Lightweight FaaS Platform for Edge Environments," in *Proceedings - 2020 IEEE International Conference on Fog Computing, ICFC 2020*, Institute of Electrical and Electronics Engineers Inc., Apr. 2020, pp. 17–24. doi: 10.1109/ICFC49376.2020.00011.

[5] J. Schleier-Smith, L. Holz, N. Pemberton, and J. M. Hellerstein, "A FaaS File System for Serverless Computing," Sep. 2020, doi: 10.48550/arXiv.2009.09845.

[6] S. Kane and K. Matthias, *Docker: Up & Running*. Accessed: Apr. 30, 2024. [Online]. Available: https://www.oreilly.com/catalog/errata.csp?isbn=9781098131821

[7] D. Von Leon, L. Miori, J. Sanin, N. El Ioini, S. Helmer, and C. Pahl, "A Lightweight Container Middleware for Edge Cloud Architectures," 2019.

[8] Y. Hao, "Edge Computing on Low Availability Devices with K3S in a Smart Home IoT System."

[9] D. Von Leon, L. Miori, J. Sanin, N. El Ioini, S. Helmer, and C. Pahl, "A Lightweight Container Middleware for Edge Cloud Architectures," 2019.

[10] P. J. R. Yepes, C. J. B. Hernandez, and L. A. Steffenel, "Impact of Containerization on Low-Cost Post Moore Computing Architectures," in *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 528–534. doi: 10.1109/CLUSTER51413.2022.00068.

[11] X. Merino and C. E. Otero, "The Cost of Virtualizing Time in Linux Containers," in *Proceedings - 2022 IEEE Cloud Summit, Cloud Summit 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 63–68. doi: 10.1109/CloudSummit54781.2022.00016.

[12] S. Kaiser, A. şaman Tosun, and T. Korkmaz, "Benchmarking Container Technologies on ARM-Based Edge Devices," *IEEE Access*, vol. 11, pp. 107331–107347, 2023, doi: 10.1109/ACCESS.2023.3321274.

[13] M. Sollfrank, F. Loch, S. Denteneer, and B. Vogel-Heuser, "Evaluating Docker for Lightweight Virtualization of Distributed and Time-Sensitive Applications in Industrial Automation," *IEEE Trans Industr Inform*, vol. 17, no. 5, pp. 3566–3576, May 2021, doi: 10.1109/TII.2020.3022843.

[14] X. I. A. Yang, W. Zhang, M. Abkar, and W. Anderson, "Computational Fluid Dynamics: its Carbon Footprint and Role in Carbon Emission Reduction," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.05985

[15] R. Nana, C. Tadonki, P. Dokladal, and Y. Mesri, "Energy Concerns with HPC Systems and Applications," Aug. 2023, [Online]. Available: http://arxiv.org/abs/2309.08615

[16] Z. Liu, F. Wu, X. Qin, C. Xie, J. Zhou, and J. Wang, "TRACER: A trace replay tool to evaluate energy-efficiency of mass storage systems," in *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 2010, pp. 68–77. doi: 10.1109/CLUSTER.2010.40.

[17] J. Panadero, S. Mendez, D. Rexachs, and E. Luque, *Characterizing Energy Efficiency in I/O system for Scientific Applications*. IARIA, 2011.