Memonitoring Network dan CPU Usage pada Virtual Machine menggunakan Prometheus dan Grafana

Rizal Maulana
Departemen Teknik Informatika dan Komputer
Program Studi Teknik Informatika
Politeknik Elektronika Negeri Surabaya
rizalmaulana@it.student.pens.ac.id

Mahargi Anugerahwan Pamungkas
Departemen Teknik Informatika dan Komputer
Program Studi Teknik Informatika
Politeknik Elektronika Negeri Surabaya
mahargip@it.student.pens.ac.id

Ezra Septian Handyanto
Departemen Teknik Informatika dan Komputer
Program Studi Teknik Informatika
Politeknik Elektronika Negeri Surabaya
ezraseptian30@it.student.pens.ac.id

Abstrak - Penelitian ini bertujuan untuk memonitoring penggunaan CPU dan jaringan di Virtual Machine menggunakan Prometheus dan Grafana yang diinstal melalui Docker. Proses monitoring mencakup pengumpulan metrik sistem dari VM menggunakan Node Exporter, pengelolaan metrik tersebut oleh Prometheus, dan visualisasi data melalui Grafana. Studi ini memberikan panduan langkah demi langkah untuk instalasi dan konfigurasi. Hasil dari implementasi ini diharapkan dapat memberikan wawasan mendalam mengenai performa CPU dan jaringan pada VM yang dimonitor.

Kata kunci - *Virtual Machine, Prometheus, Grafana, Docker, Monitoring*

1. PENDAHULUAN

1.1 Kebutuhan

Untuk memonitoring mesin virtual dengan Prometheus dan Grafana dengan docker, Anda memerlukan beberapa komponen dan konfigurasi yang tepat. Berikut adalah kebutuhan yang diperlukan:

a. Infrastruktur dan Akses

- Mesin Virtual (VM): Satu atau lebih VM yang ingin Anda monitor. Pastikan Anda memiliki akses administratif ke VM tersebut untuk menginstal dan menjalankan software.
- Docker: Menggunakan Docker untuk menjalankan server Prometheus, Grafana, dan Node Exporter membuat proses instalasi dan konfigurasi menjadi lebih sederhana dan terisolasi.

b. Perangkat Lunak yang Diperlukan

- Node Exporter: Alat untuk mengumpulkan metrik sistem dari VM.
- Prometheus: Server untuk mengumpulkan, menyimpan, dan memproses metrik.
- Grafana: Platform untuk visualisasi data metrik yang dikumpulkan oleh Prometheus.

2. RUANG LINGKUP

2.1. CPU Usage

Dalam projek ini kami melakukan monitoring penggunaan CPU secara keseluruhan dan per core, serta waktu idle. Kemudian Informasi proses yang berjalan, seperti jumlah proses, penggunaan CPU dan memori per proses. berikut penjelasan terkait mode yang akan dimonitoring :

- **idle** : Mengukur waktu CPU yang tidak digunakan untuk memproses apapun.
- **iowait** : Mengukur waktu CPU yang dihabiskan untuk menunggu operasi I/O (Input/Output) seperti disk atau jaringan selesai.
- **irq** : Mengukur waktu CPU yang digunakan untuk menangani interrupt hardware
- **nice** : Mengukur waktu CPU yang digunakan oleh proses dengan prioritas yang diubah oleh pengguna menggunakan nilai nice.
- **softirq** : Mengukur waktu CPU yang digunakan untuk menangani interrupt software.
- **steal** : Mengukur waktu CPU yang dicuri oleh hypervisor untuk keperluan menjalankan virtual CPU (vCPU) dari virtual machine lainnya.
- **system** : Mengukur waktu CPU yang digunakan oleh kernel untuk menjalankan kode kernel-space.

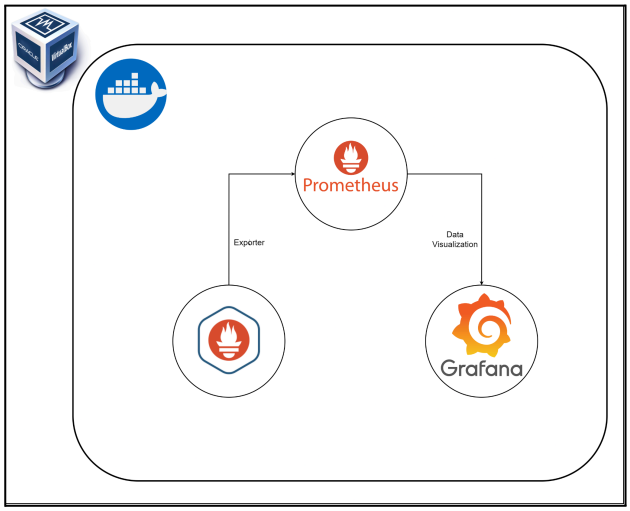
- **user** : Mengukur waktu CPU yang digunakan untuk menjalankan kode aplikasi user-space (aplikasi yang dijalankan oleh pengguna).

2.2. Network Usage

Kami juga melakukan monitoring terhadap transmit dan receive network yang berjalan di interfase network virtual machine.

3. DESAIN SISTEM

Desain sistem dan teknologi:



4. TEAM & TUGAS

Berikut adalah team members dan tugasnya:

a. Rizal Maulana
b. Mahargi Anugerahwan P
c. Ezra Septian H

Eksekutor	Task
Ezra, Maha	Install Grafana dengan Docker di virtual machine
Ezra, Maha	Install Prometheus dengan docker di virtual machine
Ezra, Rizal	Install Node Exporter di virtual machine
Ezra, Rizal	Menyambungkan node exporter dengan prometheus
Ezra	Memvisualisasikan data metriks CPU dan Network Usage yang tersimpan pada prometheus menggunakan grafana
Rizal, Maha, Ezra	Membuat laporan akhir

5. WAKTU PELAKSANAAN

Waktu	Task
27 Mei 2024	Membuat Project Charter
28 Mei 2024	Menyiapkan Prometheus dan Grafana yang berjalan di Virtual Machine
29 Mei - 2 Juni 2024	Menyambungkan metrics dari prometheus ke grafana & Memvisualisasikan data di grafana
3 Juni 2024	Menyusun dan mengumpulkan kesimpulan

6. SUMBER

Medium :

- https://medium.com/@dedihartono.drive/basic-prometheus-untuk-monitoring-dan-alerting-di-vm-virtual-machine-9a3a3088dd22
- https://medium.com/@extio/unveiling-the-architectural-brilliance-of-prometheus-af07cca14896

Dokumentasi :

- https://docs.techdox.nz/prometheus/
- https://docs.techdox.nz/grafana/
- https://docs.techdox.nz/node-exporter/

7. IMPLEMENTASI

1. Install Grafana dengan Docker di Virtual Machine
    a. Pastikan Docker engine dan composer sudah terinstall, dan network pada virtual Machine diarahkan ke Bridge Adapter (menggunakan Wifi) agar mendapatkan IP address. Setelah itu lakukan cd ~ kemudian buat repository baru dengan mkdir Docker, setelah itu buat 2 repository baru yaitu grafana dan prometheus.
    b. Setelah membuat repository masuk ke repository grafana dan buat file baru menggunakan sudo nano docker-compose.yml, setelah itu paste ini ke dalam file tersebut:

```
version: "3.8"
services:
  grafana:
    image: grafana/grafana
    container_name: grafana
    restart: unless-stopped
    ports:
     - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana
volumes:
  grafana-storage: {}
```

version: "3.8": Menggunakan versi Docker Compose 3.8.

services: Bagian ini mendefinisikan layanan yang akan dijalankan.

grafana: Definisi layanan untuk Grafana.

image: grafana/grafana: Menggunakan gambar Docker untuk Grafana.

container_name: grafana: Memberikan nama grafana untuk kontainer.

restart: unless-stopped: Kebijakan restart kontainer (restart kecuali dihentikan secara manual).

ports: Mendefinisikan pemetaan port.

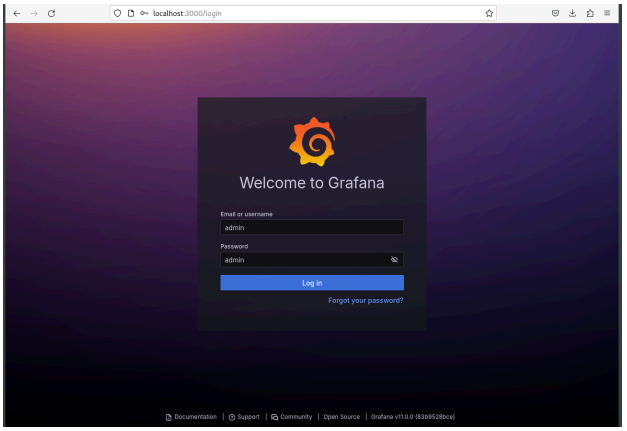
'3000:3000': Memetakan port 3000 di host ke port 3000 di kontainer.

volumes: Mendefinisikan volume untuk penyimpanan data persisten.

grafana-storage:/var/lib/grafana: Menggunakan volume grafana-storage yang dipetakan ke /var/lib/grafana di dalam kontainer.

grafana-storage: {}: Definisi volume dengan nama grafana-storage.

    c. Setelah itu lakukan perintah pada terminal *docker compose up -d*
    d. Kemudian tunggu hingga selesai, jika sudah buka browser dan masukkan url *localhost:3000* dan akan muncul tampilan grafana seperti ini yang menunjukkan grafana berhasil diinstall. Untuk username dan password pertama kali yaitu *admin* dan passwordnya *admin*.



2. Install prometheus dengan Docker di Virtual Machine

a. Buka folder prometheus yang sudah kita buka.
b. Buat file docker-compose.yml dan folder prometheus, kemudian di dalam folder tersebut tambahkan file promotheus.yml
c. Masukkan code berikut ke dalam prometheus.yml:

```
global:
  scrape_interval: 15s
  scrape_timeout: 10s
  evaluation_interval: 15s
alerting:
  alertmanagers:
    - static_configs:
      - targets: []
      scheme: http
      timeout: 10s
      api_version: v1
scrape_configs:
  - job_name: prometheus
    honor_timestamps: true
    scrape_interval: 15s
    scrape_timeout: 10s
    metrics_path: /metrics
    scheme: http
    static_configs:
      - targets:
        - localhost:9090
  - job_name: elzim # Change to whatever
          you like
    static_configs:
      - targets: ['192.168.45.62:9100']
```

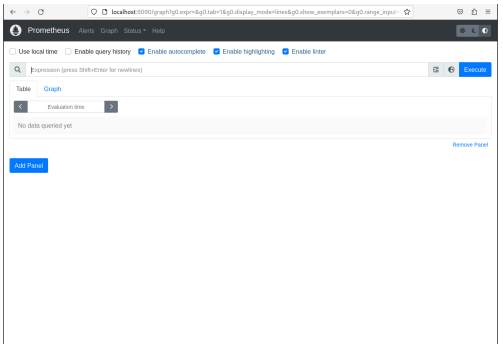
Kemudian untuk targets: ['192.168.45.62:9100'] sesuaikan dengan services node exporter (install node exporter setelah prometheus).

d. Setelah itu masukkan code berikut ke dalam docker-compose.yml yang ada di dalam repository Docker/prometheus

```
version: '3.8'
services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    command:
      -
'--config.file=/etc/prometheus/prometheu
s.yml'
    ports:
      - 9090:9090
    restart: unless-stopped
    volumes:
      - ./prometheus:/etc/prometheus
      - prom_data:/prometheus
volumes:
  prom_data:
```

e. Setelah itu jalankan *docker compose up -d* dan tunggu hingga selesai.

f. Jika sudah selesai coba lakukan hal yang sama seperti grafana tetapi dengan port 9090 dan akan muncul seperti ini.



3. Install Node Exporter di Virtual Machine

a. Ketikkan cd ~ kemudian download file node_exporter-1.8.1.linux-amd64.tar.gz dengna cara wget https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz setelah itu tunggu hingga download selesai.
b. Setelah selesai download, extract file tersebut dengan cara tar xvf node_exporter-1.8.1.linux-amd64.tar.gz.
c. Kemudian buka repository node_exporter-1.8.1.linux-amd64 dan move repository node_exporter ke /usr/local/bin dengan cara sudo mv node_exporter /usr/local/bin pada terminal.
d. Setalah itu hapus repository node_exporter-1.8.1.linux-amd64 dengan cara cd .. kemudian sudo rm node_exporter-1.8.1.linux-amd64
e. Kemudian ketikkan sudo useradd --no-create-home --shell /bin/false node_exporter dan sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
f. Setelah itu ketikkan sudo nano /etc/systemd/system/node_exporter.service dan isi dari file tersebut:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
```

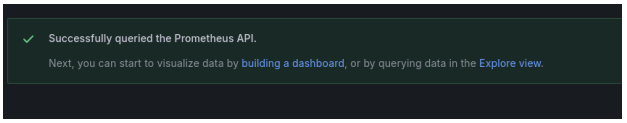
g. Setelah itu ketikkan sudo systemctl daemon-reload, sudo systemctl enable node_exporter dan sudo systemctl start node_exporter
h. jika sudah cek apakah node_exporter berjalan dengan cara sudo systemctl status node_exporter.service. Jika berjalan maka akan muncul seperti ini:

```
ezra@SysAdmin-3122600016:~$ sudo systemctl daemon-reload
ezra@SysAdmin-3122600016:~$ sudo systemctl enable node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/no
de_exporter.service.
ezra@SysAdmin-3122600016:~$ sudo systemctl start node_exporter
ezra@SysAdmin-3122600016:~$ sudo systemctl status node_exporter.service
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-06-02 19:55:55 WIB; 8s ago
   Main PID: 14439 (node_exporter)
      Tasks: 4 (limit: 3633)
     Memory: 14.6M
        CPU: 27ms
     CGroup: /system.slice/node_exporter.service
             └─14439 /usr/local/bin/node_exporter
```

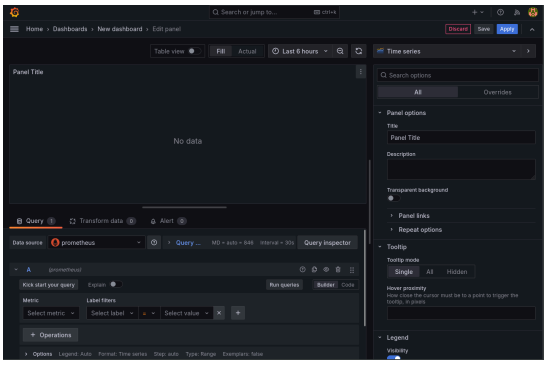
i. Setelah itu cek pada prometheus pada bagian status -> targets apakah 9100/metrics sudah state up, jika sudah maka prometheus sudah dapat menyimpan metriks metriks yang dikirim oleh node_exporter.
j. matriks-matriks yang didapat oleh NodeExporter untuk di terus kan ke prometheus dari file yang ada difolder /proc/system/** yang menyimpan data terkait CPU dan Network Usage

4. Membuat Tampilan Dashboard menggunakan Grafana dengan resource dari Prometheus berupa metriks metriks yang didapat dari node_exporter

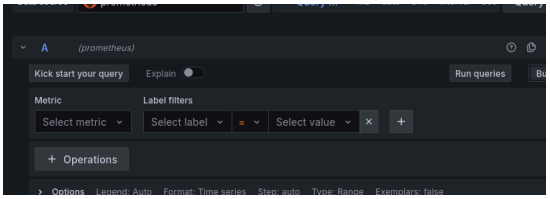
a. Buka grafana dan login sebagai admin, setelah itu pada tampilan home klik pada bagian Data Sources dan pilih prometheus, dan untuk prometheus server url isi dengan http://192.168.45.62:9090 (sesuaikan ip) dan klik Save & test. Jika berhasil akan muncul seperti ini



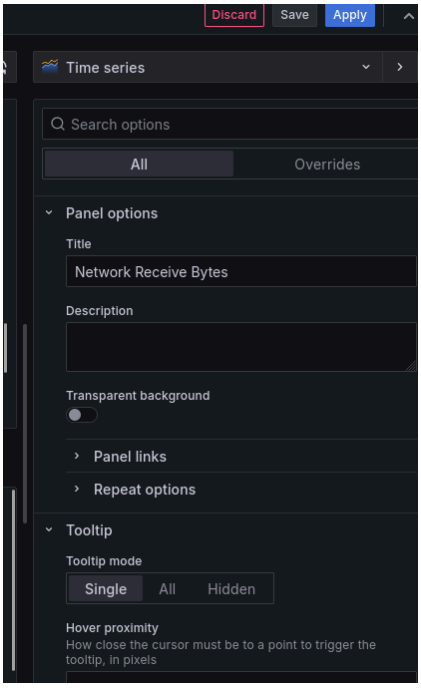
b. Setelah itu pada tampilan home page, klik ‘create your first dashboard’, kemudian *add visualization*, lalu pilih prometheus



c. kemudian setting metric node_cpu_second_total untuk CPU Usage dan node_network_receive_bytes_total untuk Network Usage, lalu tambahkan operations rate di range functions, set rangenya 1m, kemudian jalankan run query

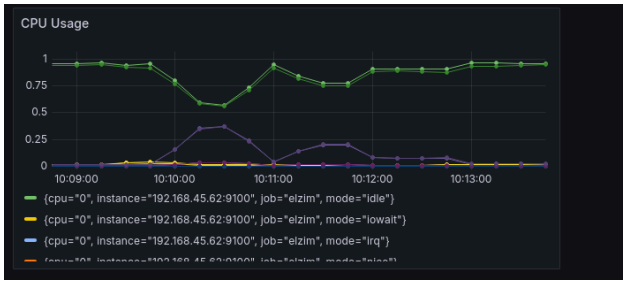


d. setelah itu beri nama panel sesuai data metriks yang dimonitoring dan klik apply untuk menambahkan ke dashboard

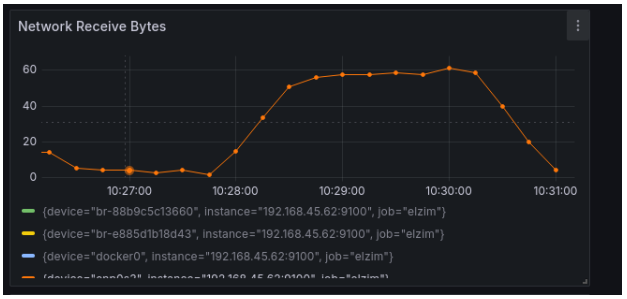


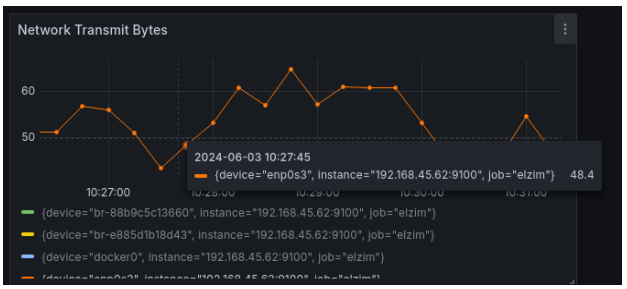
e. ditampilan dashboard akan muncul panel yang sudah kita buat tadi, jika ingin menambah panel bisa klik add > visualization

8. ANALISA



Dapat dilihat bahwa penggunaan cpu dominan pada mode idle karena tidak adanya proses yang membebani CPU, jika terdapat proses yang membebani CPU maka CPU dengan mode user akan naik penggunannya karena mode user mencakup sebagian besar aplikasi pengguna dan program non-sistem dijalankan.





Untuk melihat network transmit byte dan network receive byte bisa dipilih interface yang sesuai dengan network kita, untuk mendapatkan interface sesuai dengan network kita, kita dapat menggunakan command ip route show dan pilih ip 192.168.xx.0 kemudian lihat interfacenya. Disini kita menggunakan enp0s3.

Pengaruh CPU usage terhadap Network Usage: Penggunaan CPU dan penggunaan jaringan (network usage) tidak selalu berkaitan langsung, tetapi dalam beberapa kasus, keduanya bisa saling mempengaruhi. Misalnya, saat CPU bekerja keras untuk menangani tugas-tugas berat, bisa jadi penggunaan jaringan terpengaruh dengan penurunan throughput. Namun, dampaknya bisa berbeda tergantung pada jenis aktivitas dan kondisi sistem yang sedang berjalan.

CPU Benchmarking:

```
ezra@SysAdmin-3122600016:~/flops-iops$ iops32
Number of CPU cores to run Benchmark: 1
Benchmarking for 32 Bit Integer operations per second
1| Tr 1: 5185962985 IOPS = 5185962985
Maximum CPU Throughput: 5.185963 Gigaiops.
Maximum Single Core Throughput: 5.185963 Gigaiops.
ezra@SysAdmin-3122600016:~/flops-iops$ iops64
Number of CPU cores to run Benchmark: 2
Benchmarking for 64 Bit Integer operations per second
1| Tr 1: 5796977472 Tr 2: 5738173298 IOPS = 11535150770
Maximum CPU Throughput: 11.535151 Gigaiops.
Maximum Single Core Throughput: 5.796978 Gigaiops.
ezra@SysAdmin-3122600016:~/flops-iops$ iops64
Number of CPU cores to run Benchmark: 1
Benchmarking for 64 Bit Integer operations per second
1| Tr 1: 5946541900 IOPS = 5946541900
Maximum CPU Throughput: 5.946542 Gigaiops.
ezra@SysAdmin-3122600016:~/flops-iops$ iops32
Number of CPU cores to run Benchmark: 2
Benchmarking for 32 Bit Integer operations per second
1| Tr 1: 4269443239 Tr 2: 3466499435 IOPS = 7735942674
Maximum CPU Throughput: 7.735943 Gigaiops.
Maximum Single Core Throughput: 4.269444 Gigaiops.
```

```
ezra@SysAdmin-3122600016:~/flops-iops$ flops32
Number of CPU cores to run Benchmark: 1
Benchmarking for 32 Bit Floating point operations per second
1| Tr 1: 4818243314 FLOPS = 4818243314
Maximum CPU Throughput: 4.818243 Gigaflops.
Maximum Single Core Throughput: 4.818243 Gigaflops.
```

```
ezra@SysAdmin-3122600016:~/flops-iops$ flops32
Number of CPU cores to run Benchmark: 2
Benchmarking for 32 Bit Floating point operations per second
1| Tr 1: 3880985271 Tr 2: 3467941351 FLOPS = 7348926622
Maximum CPU Throughput: 7.348927 Gigaflops.
Maximum Single Core Throughput: 3.880985 Gigaflops.
```

```
ezra@SysAdmin-3122600016:~/flops-iops$ flops64
Number of CPU cores to run Benchmark: 1
Benchmarking for 64 Bit Floating point operations per second
1| Tr 1: 4536850552 FLOPS = 4536850552
Maximum CPU Throughput: 4.536850 Gigaflops.
Maximum Single Core Throughput: 4.536850 Gigaflops.
```

```
ezra@SysAdmin-3122600016:~/flops-iops$ flops64
Number of CPU cores to run Benchmark: 2
Benchmarking for 64 Bit Floating point operations per second
1| Tr 1: 5286520798 Tr 2: 5222402874 FLOPS = 10508923672
Maximum CPU Throughput: 10.508924 Gigaflops.
Maximum Single Core Throughput: 5.286521 Gigaflops.
```

"FLOPS" adalah singkatan dari "Floating Point Operations Per Second" dan merupakan satuan yang digunakan untuk mengukur kinerja atau kecepatan komputer dalam melakukan operasi aritmatika floating point per detik. Ini sering digunakan sebagai ukuran kinerja dalam komputasi ilmiah dan aplikasi yang membutuhkan banyak perhitungan matematika, seperti rendering grafis, simulasi fisika, dan analisis data numerik.

"IOPS" adalah singkatan dari "Input/Output Operations Per Second" dan merupakan satuan yang digunakan untuk mengukur jumlah operasi input/output yang dapat dilakukan oleh sebuah perangkat (seperti hard drive, solid-state drive, atau jaringan) dalam satu detik. Ini merupakan indikator kinerja dalam hal kemampuan perangkat untuk memproses permintaan baca/tulis data dalam rentang waktu tertentu. IOPS penting dalam aplikasi yang membutuhkan akses data yang cepat, seperti basis data atau sistem penyimpanan berkinerja tinggi.

Kemudian untuk output pada bagian 1 | Tr 1: Ini menunjukkan bahwa pada core pertama (Tr 1), CPU mampu melakukan sekian operasi floating point per detik/input output per detik, atau sekitar sekian FLOPS atau IOPS.

Maximum CPU Throughput: sekian Gigaflops /Gigaiops: Ini menunjukkan throughput (kapasitas maksimum) total CPU dalam melakukan operasi floating point per detik atau input output per detik, dalam hal ini sekitar sekian Gigaflops atau Gigaiops

Maximum Single Core Throughput: sekian Gigaflops/Gigaiops: Ini menunjukkan throughput (kapasitas maksimum) dari satu core CPU dalam melakukan operasi floating point per detik atau input output per detik, dalam hal ini sekitar Gigaflops/Gigaiops.

9. KESIMPULAN

Pelaksanaan workshop ini berhasil mencapai tujuan utama yaitu memonitoring penggunaan CPU dan jaringan pada Virtual Machine. Melalui instalasi dan konfigurasi Docker, Prometheus, dan Grafana, kita dapat mengumpulkan, menyimpan, dan memvisualisasikan data metrik secara efisien. Penggunaan Node Exporter terbukti efektif dalam mengumpulkan metrik sistem, sementara Prometheus mampu mengelola dan memproses data tersebut dengan baik. Grafana memberikan antarmuka yang intuitif untuk memvisualisasikan metrik yang terkumpul, sehingga memudahkan analisis performa VM. Implementasi ini menunjukkan bahwa pendekatan menggunakan Docker untuk menjalankan komponen-komponen monitoring dapat menyederhanakan proses dan meningkatkan isolasi sistem, memberikan model yang dapat direplikasi untuk kebutuhan monitoring serupa di lingkungan lain.