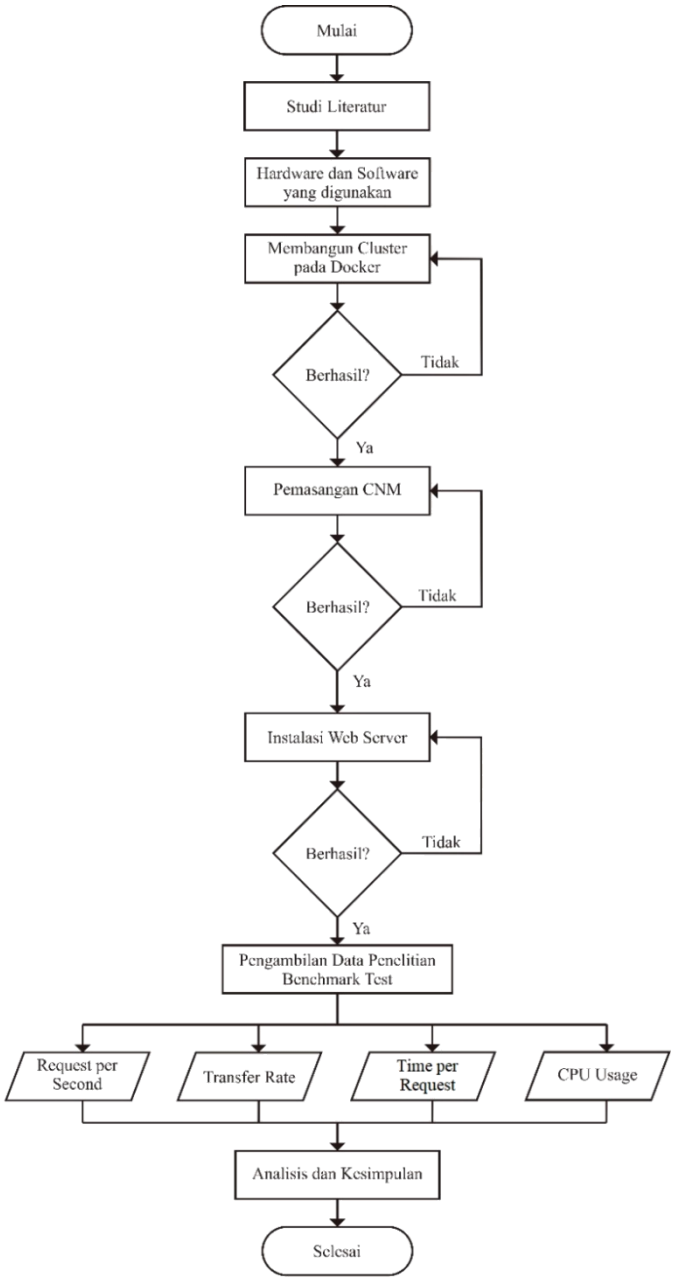
BAB 3
METODE PENELITIAN

3.1 ALUR PENELITIAN

Penelitian yang dilakukan terdiri dari beberapa tahapan, mulai dari studi literatur hingga tahap penyusunan laporan. Gambar 3.1 adalah diagram alur yang digunakan pada penelitian ini.



Gambar 3.1 Alur Penelitian

3.2 STUDI LITERATUR

Studi literatur menjadi tahap pertama dalam proses penelitian yang bertujuan untuk mencari referensi untuk dijadikan sebagai acuan dalam pelaksanaan penelitian. Sumber yang digunakan oleh penulis berupa jurnal, buku, dan *website* yang berhubungan dengan topik penelitian.

3.3 *HARDWARE* DAN *SOFTWARE* YANG DIGUNAKAN

Perangkat-perangkat yang digunakan untuk membuat simulasi dalam penelitian ini terdiri atas perangkat keras (*hardware*) dan perangkat lunak (*software*) untuk *manager node*, *web server*, dan *benchmark tools*. Spesifikasi *hardware* dan *software* yang digunakan ditunjukkan pada Tabel 3.1.

Tabel 3.1 Spesifikasi *Hardware* dan *Software* yang Digunakan

Perangkat	Spesifikasi	Keterangan
Manager Node dan *Worker Node*	OS	*Ubuntu Server* 20.04
	CPU	2 vCPU
	RAM	8 GB
	Tool dan Aplikasi	1. *Docker* 20.10.19 2. CNM *Bridge Network* 3. CNM *Overlay Network* 4. CNM *Weave Net*
Web Server	*Version*	*Apache* 2.4.54
Benchmark tools	*Apps*	*Apache Benchmark* 2.4 *Htop* 2.2.0

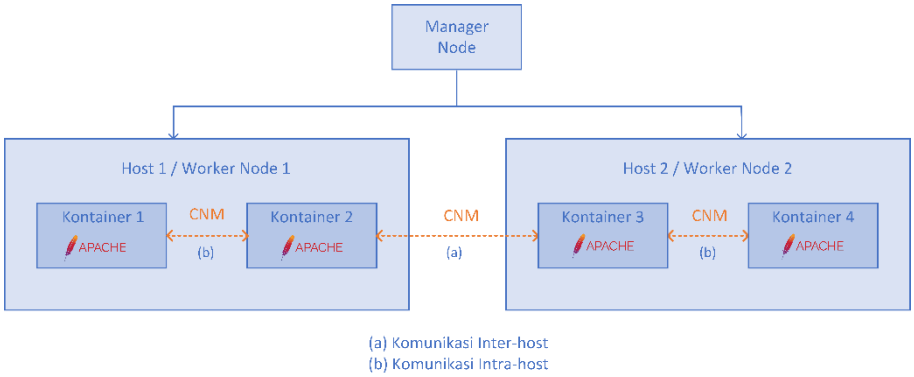
3.4 PERANCANGAN SKENARIO

Skenario yang digunakan pada penelitian ini yaitu penggunaan empat CNM yang berbeda yaitu *Overlay Network*, *Bridge Network*, dan *Weave Net* pada komunikasi *intra-host* dan komunikasi *inter-host*. Masing-masing skenario komunikasi dilakukan percobaan CNM secara bergantian dengan percobaan sebanyak 30 kali. Skenario penelitian dimaksudkan untuk mengetahui performansi penggunaan CNM yang berbeda-beda untuk trafik *web server*. Pada Tabel 3.2 berisi skenario yang digunakan dalam penelitian.

Tabel 3.2 Skenario Penelitian

Skenario	CNM	Jenis Komunikasi	Jumlah Koneksi	Jumlah *Request*	Jumlah Percobaan
1	*Overlay Network*	komunikasi *intra-host*	100	100	30
		komunikasi *inter-host*			
2	*Bridge Network*	komunikasi *intra-host*	100	100	30
		komunikasi *inter-host*			
3	*Weave Net*	komunikasi *intra-host*	100	100	30
		komunikasi *inter-host*			

Skenario pertama yang digunakan pada penelitian ini yaitu penggunaan CNM *Overlay Network* dengan komunikasi *intra-host* dan komunikasi *inter-host*. Skenario pertama ini bermaksud untuk mengetahui performansi CNM *Overlay Network* yang digunakan terhadap parameter yang diamati yaitu *request per second, transfer rate, time per request*, dan CPU *usage*. Begitu juga dengan skenario kedua dan ketiga, penggunaan CNM *Bridge Network* dan *Weave Net* dengan komunikasi sama yaitu komunikasi *intra-host* dan komunikasi *inter-host.* Masing-masing skenario menggunakan koneksi berjumlah 100 *user* dan *request* berjumlah 100 permintaan pada setiap *user*. Sehingga total beban *benchmark* adalah 10.000 *request*, dengan total tersebut sudah dapat dikatakan hampir seperti permintaan akses ke *web server* di dunia nyata.



Gambar 3.2 Skenario Komunikasi pada *Docker Swarm Cluster*

Skenario komunikasi *intra-host* dan komunikasi *inter-host* digambarkan pada Gambar 3.2. Komunikasi *intra-host* merupakan komuikasi antar kontainer yang berada dalam satu *host* atau *worker node*, sedangkan komunikasi *inter-host* merupakan komunikasi antar kontainer yang berada pada *host* atau *worker node* yang berbeda. Setiap container akan dihubungkan menggunakan CNM yang diuji secara bergantian, yaitu pertama CNM *Overlay Network,* kedua CNM *Bridge Network,* dan terakhir CNM *Weave Net.*

3.5 PARAMETER PENELITIAN

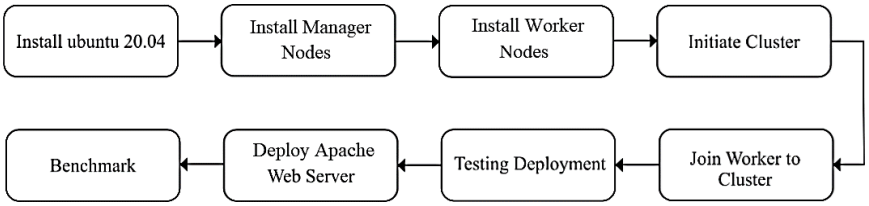
Parameter *benchmark* yang dianalisis pada penelitian ini yaitu *requests per second, transfer rate, time per request,* dan CPU *usage*. Hasil dari parameter tersebut didapatkan menggunakan *tool Apache Benchmark* dan *Htop*. Pengujian ini dilakukan dengan menguji beban request trafik *Web Server Apache*. Trafik *web server* yang digunakan berupa trafik HTTP yang berjalan pada *transport layer* yaitu TCP dan UDP dalam protokol TCP/IP. Tabel 3.3 adalah tabel parameter yang diamati pada penelitian.

Tabel 3.3 Parameter Pengujian

Parameter Penelitian	Satuan
Requests per second	*request/s*
Transfer rate	*KBps*
Time per request	ms
CPU *Usage*	%

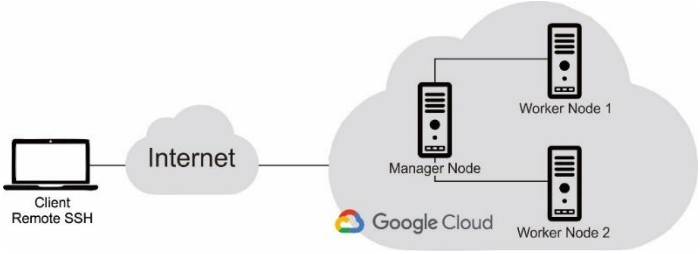
3.6 PROSES SIMULASI

Pada tahap ini, akan dilakukan pembuatan simulasi yang dijalankan dengan skenario yang dirancang seperti pada Gambar 3.3. Langkah awal yaitu mempersiapkan *hardware* dan *software* yang digunakan untuk proses simulasi. Pada proses simulasi pertama menginstal sistem operasi *Ubuntu Server* 20.04 yang digunakan sebagai *server*. Selanjutnya menginstal *Docker* pada *server* dan membuat *cluster* yang terdiri dari *manager node* dan *worker node*. Mengaktifkan *Docker Swarm* untuk melakukan orkestrasi kontainer pada *Docker*. Melakukan instalasi CNM *Bridge Network, Overlay Network, dan Weave Net* secara bergantian untuk setiap kali pengujian. Tahapan selanjutnya yaitu membuat *deployment* aplikasi *web server Apache* untuk melakukan *test benchmark*. *Benchmark* dilakukan menggunakan *tool Apache Benchmark* dan *Htop* untuk mendapatkan hasil dari parameter yang diuji. Selain itu terdapat proses pemodelan sistem dilakukan proses *setup Google Cloud Platform*, instalasi *Docker, Docker Swarm, Container Network Model* (CNM), serta *web server Apache*.



Gambar 3.3 Diagram Blok Sistem

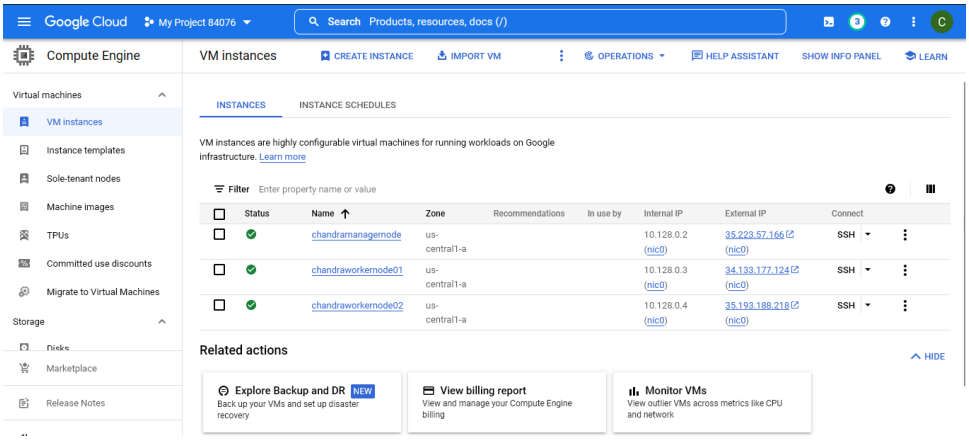
Pada gambar 3.4 merupakan gambar topologi jaringan yang digunakan. Topologi jaringan dibagi benjadi dua segmen yaitu bagian *client* yang akan melakukan remot SSH kepada *server* melalui internet, dan juga bagian *server* yang berada di *google cloud*. *Server* yang ada di *google cloud* akan dibuat menjadi *cluster* dari *Docker Swarm* yang dibangun sebagai orkestrasi kontainer. *Docker Swarm cluster* tersusun dari satu *manager node* dan dua *worker node*.



Gambar 3.4 Topologi Jaringan

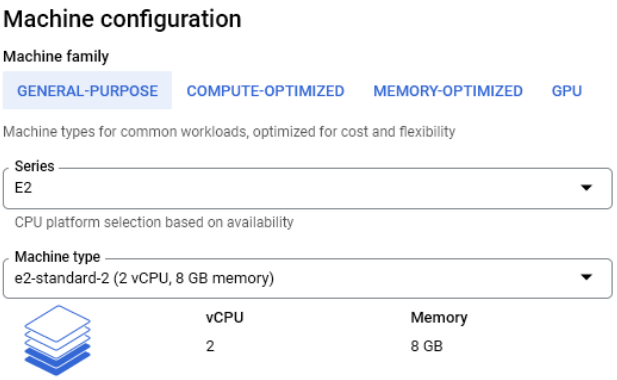
3.6.1 Setup *Google Cloud Platform*

Google Cloud Platform atau yang lebih dikenal dengan GCP merupakan suatu layanan dari *Google* yang menyediakan berbagai kebutuhan *server* berbasis *cloud*. Penelitian ini menggunakan fitur VM *instances* yaitu menggunakan 3 *server* untuk proses simulasi seperti pada gambar 3.5.



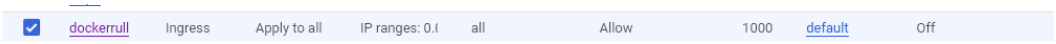
Gambar 3.5 *Virtual Machine Instances Google Cloud Platform*

Spesifikasi yang digunakan pada ketiga *server* seperti pada gambar 3.6 yaitu menggunakan 2vCPU dan 8GB memori dengan menggunakan sistem operasi Ubuntu 20.04. Konfigurasi yang ditambahkan ketika membuat *server* ini yaitu menambahkan SSH-*key* yang telah dibuat agar ketiga *server* tersebut dapat diakses secara aman oleh *user*.



Gambar 3.6 Spesifikasi *Server*

Penggunaan *Docker* pada *Google Cloud Platform* harus membuat sebuah *firewall* khusus yang digunakan untuk membuka akses komunikasi pada *port* yang digunakan untuk *Docker Swarm* seperti pada gambar 3.7. *Firewall* tersebut dikonfigurasi agar seluruh trafik *ingress* diijinkan untuk melewati GCP pada *range IP* 0.0.0.0/0.



Gambar 3.7 *Firewall GCP*

3.6.2 Instalasi *Docker* dan *Docker Swarm*

Pemasangan *Docker* memiliki beberapa tahapan dimulai dari pemetaan alamat *IP cluster host*, *docker engine*, dan yang terakhir yaitu instalasi *Docker Swarm*. Pada tahapan yang pertama yaitu memetakan nama *host* dan alamat *IP manager node*, 1 *worker node*, dan 2 *worker node* ke dalam *directory* /etc/hosts. Konfigurasi ini dilakukan pada seluruh *host.*

```
brica@chandramanagernode:~$ sudo nano /etc/hosts
10.128.0.2 chandramanagernode
10.128.0.3 chandraworkernode01
10.128.0.4 chandraworkernode02
```

Selanjutnya melakukan instalasi *docker engine* dengan terlebih dahulu melakukan *update package* Ubuntu.

```
brica@chandramanagernode:~$ sudo apt -y update
brica@chandramanagernode:~$ sudo apt -y install apt-transport-https ca-
certificates curl gnupg-agent software-properties-common
```

Perintah diatas merupakan perintah instalasi paket prasyarat yang diperlukan untuk *Docker*.

```
brica@chandramanagernode:~$ sudo curl -fsSL
https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Pada langkah selanjutnya, menambahkan repositori *Docker* resmi ke sistem Ubuntu 20.04. Melakukan pembaruan indeks paket lokal untuk membuat sistem mengetahui repositori yang baru ditambahkan.

```
brica@chandramanagernode:~$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Kemudian instal *Docker* dari repositori *Docker* resmi. Perintah dibawah untuk menginstal *Docker* bersama paket tambahan yang akan dibutuhkan oleh *Docker*.

```
brica@chandramanagernode:~$ sudo apt -y install docker-ce docker-ce-cli
containerd.io
brica@chandramanagernode:~$ sudo systemctl status docker
```

Mengecek apakah *Docker* sudah terinstal dan berstatus aktif atau tidak. Setelah *Docker* diinstal, tambahkan *user* yang saat ini masuk ke grup *Docker*.

```
brica@chandramanagernode:~$ sudo usermod -aG docker ${USER}
brica@chandramanagernode:~$ newgrp docker
```

Perintah diatas adalah untuk menghindari menjalankan *Docker* sebagai *user* sudo setiap kali menjalankan *Docker*.

Proses penginstalan *Docker Swarm* dilakukan hanya pada *manager node* saja dengan perintah bawah.

```
brica@chandramanagernode:~$ sudo docker swarm init --advertise-addr
10.128.0.2
```

Langkah selanjutnya adalah menginisialisasi *Docker Swarm pada manager node*. Setelah *Docker Swarm* diinisialisasi, kemudian menambahkan atau menggabungkan *worker node* ke *cluster* untuk membuat *Docker Swarm Cluster* dan akan ditampilkan di terminal.

```
brica@chandramanagernode:~$ docker swarm join --token SWMTKN-1-
1yuh7l9l96sorhiyydmgyhcgdn4xmhlu66hbubx0nacrditi65-
1kemy1qii3hgz02amhjpsjplt 10.128.0.2:2377
```

Menjalankannya di setiap *worker node* seperti yang dibuat sebelumnya. Selanjutnya *login* kembali ke masing-masing *worker node* dan tambahkan perintah untuk bergabung dengan *cluster*.

```
brica@chandramanagernode:~$ docker node ls
```

Melakukan pengecekan apakah seluruh *node* sudah tergabung ke dalam *cluster Docker Swarm* seperti pada gambar 3.8.

```
brica@chandramanagernode:~$ docker node ls
ID                            HOSTNAME              STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
hx4wr7tyryh7i9dzjp42s2d16 *   chandramanagernode    Ready     Active         Leader           20.10.21
rmhc896mx48x1hl8oa71oip95     chandraworkernode01   Ready     Active                          20.10.21
9qe384qfh62j11uau96wiz0k8     chandraworkernode02   Ready     Active                          20.10.21
```

Gambar 3.8 Daftar Docker *node*

3.6.3 Instalasi *Container Network Model* (CNM)

Instalasi CNM pada *Docker Warm* dilakukan secara bergantian untuk setiap simulasinya. Sehingga akan dilakukan tiga kali pemasangan CNM yaitu *bridge network*, *overlay network*, dan juga *weave net*, seperti pada gambar 3.9.

```
brica@chandramanagernode:~$ docker network ls
NETWORK ID     NAME              DRIVER                                  SCOPE
ffe50268122a   bridge            bridge                                  local
0b046d2bec80   docker_gwbridge   bridge                                  local
d626d337ea2b   host              host                                    local
94c155f20171   none              null                                    local
dqaff7fcugtr   overlay           overlay                                 swarm
lnumn6qz19fs   weavenet          weaveworks/net-plugin:latest_release    swarm
```

Gambar 3.9 Daftar CNM pada *Docker*

A. *Bridge Network*

Instalasi CNM *bridge network* pada *manager node* dengan perintah berikut.

```
brica@chandramanagernode:~$ docker network create --driver bridge bridge
brica@chandramanagernode:~$ docker network ls
```

B. *Overlay Network*

Instalasi CNM *overlay network* pada *manager node* dengan perintah berikut.

```
brica@chandramanagernode:~$ docker network create --driver overlay overlay
brica@chandramanagernode:~$ docker network ls
```

C. *Weave Net*

Instalasi CNM *weave net* pada *manager node* dengan perintah berikut.

```
brica@chandramanagernode:~$ docker plugin install weaveworks/net-plugin:latest_release
brica@chandramanagernode:~$ docker plugin disable weaveworks/net-plugin:latest_release
brica@chandramanagernode:~$ docker plugin set weaveworks/net-plugin:latest_release PARAM=VALUE
brica@chandramanagernode:~$ docker plugin enable weaveworks/net-plugin:latest_release
```

Instalasi CNM *weave net* sedikit berbeda dari instalasi CNM *bridge* dan *overlay*. *Plugin weave net* terlebih dahulu ditambahkan ke dalam *Docker* untuk seluruh *host* baik *manager node* dan *worker node*. Selanjutnya dilakukan pemasangan CNM *weave net* pada *manager node* saja.

```
brica@chandramanagernode:~$ docker network create --
driver=weaveworks/net-plugin:latest_release weavenet
```

3.6.4 Instalasi *Web Server Apache*

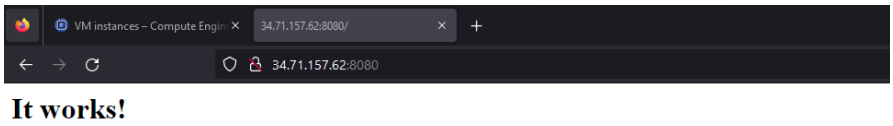
Instalasi *Apache web server* pada *Docker Swarm* yaitu menggunakan *image httpd* dengan versi *apache web server* 2.4. *Service Apache* akan diinstal menggunakan ketiga jenis CNM secara bergantian dengan menggunakan 4 *replicas* atau 4 kontainer *node web server Apache*. *Service Apache* akan di ekspose pada *port* 8080:80 seperti pada gambar 3.10.

brica@chandramanagernode:~$ docker service create --name apache --network bridge --replicas 4 -p 8080:80 httpd:2.4

```
brica@chandramanagernode:~$ docker service create --name apache --network bridge --replicas 4 -p 8080:80 httpd:2.4
m3exzgg4zyfrbo8a7wmes8tya
overall progress: 4 out of 4 tasks
1/4: running   [==================================================>]
2/4: running   [==================================================>]
3/4: running   [==================================================>]
4/4: running   [==================================================>]
verify: Service converged
```

Gambar 3.10 *Install service web server Apache*

Setelah instalasi *web server Apache* selesai dilakukan, maka *web server* sudah dapat diakses melalui IP *host worker node* yang tersedia pada *Google Cloud Platform*. Gambar 3.11 menampilkan halaman *web server Apache* yang sudah dapat diakses.



Gambar 3.11 Halaman *web server Apache*

3.7 PENGAMBILAN DATA PENELITIAN

Data yang diamati untuk dianalisis yaitu parameter *benchmark* yang terdiri dari *requests per second, transfer rate, time per request*, dan CPU *usage* dari skenario komunikasi *intra-host* dan komunikasi *inter-host* sebanyak 30 kali percobaan. Data tersebut diambil dengan *tools Apache Benchmark* dan *Htop* yang diinstal pada *container*.

3.7.1 *Apache Benchmark*

Pengambilan data *request per second, transfer rate,* dan *time per request* menggunakan aplikasi *Apache Benchmark* dilakukan pada dua jenis komunikasi yaitu *inter-host* dan *intra-host*. Aplikasi *Apache Benchmark* diinstal pada salah satu kontainer yang berada pada *worker node* dengan perintah dibawah. *User root* didapatkan secara *generate* otomatis pada *worker node* sehingga tidak dapat diubah namanya.

```
root@7d26a62aff31:/usr/local/apache2# apt install apache2-utils
```

Proses pengambilan data dilakukan pada dua skenario komunikasi. Untuk komunikasi *inter-host*, pengambilan data dilakukan pada kontainer *worker node1* yang mengakses *web server* pada alamat IP *worker node1* pada *host* yang sama dengan perintah berikut.

```
root@7d26a62aff31:/usr/local/apache2# ab -n 100 -c 100 -g bridgeintrahost01.txt http://34.133.177.124:8080/
```

Sedangkan proses pengambilan data pada komunikasi *intra-host*, pengambilan data dilakukan pada kontainer *worker node1* yang mengakses *web server* pada alamat IP *worker node2* pada *host* yang bebeda dengan perintah berikut.

```
root@7d26a62dvc54:/usr/local/apache2# ab -n 100 -c 100 -g bridgeintrahost01.txt http://34.116.113.90:8080/
```

Kode ab berarti untuk menggunakan aplikasi *Apache Benchmark*, -n 100 berarti jumlah *request* atau permintaan, -c 100 berarti jumlah *connection* atau koneksi, -g berarti *gnuplot-file* untuk menyimpan hasil dari *benchmark* ke dalam *file* berformat txt, serta http://34.133.177.124:8080/ berarti untuk mengakses IP *worker node* dengan *port* 8080.

Secara otomatis aplikasi *Apache Benchmark* akan menampilkan hasil nilai *benchmark* parameter *request per second*, *time per request*, dan *transfer rate*. Selain ketiga parameter tersebut, *Apache Benchmark* juga menampilkan hasil dari parameter lainnya yang dapat digunakan untuk nilai pada perhitungan manual menggunakan rumus, seperti ditunjukkan pada gambar 3.12.

```
root@7d26a62aff31:/usr/local/apache2# ab -n 100 -c 100 -g bridgeinterhost10.txt http://35.193.188.218:8080/
This is ApacheBench, Version 2.3 <$Revision: 1901567 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 35.193.188.218 (be patient).....done


Server Software:        Apache/2.4.54
Server Hostname:        35.193.188.218
Server Port:            8080

Document Path:          /
Document Length:        45 bytes

Concurrency Level:      100
Time taken for tests:   0.034 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      28900 bytes
HTML transferred:       4500 bytes
Requests per second:    2949.50 [#/sec] (mean)
Time per request:       33.904 [ms] (mean)
Time per request:       0.339 [ms] (mean, across all concurrent requests)
Transfer rate:          832.43 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        2    8   1.8      9      11
root@7d26a62aff31:/usr/local/apache2#
Waiting:        1    9   4.8      8      23
Total:         11   18   4.3     17      30
```

Gambar 3.12 Pengambilan data *request per second, time per request*, dan *transfer rate* dengan *Apache benchmark tools*

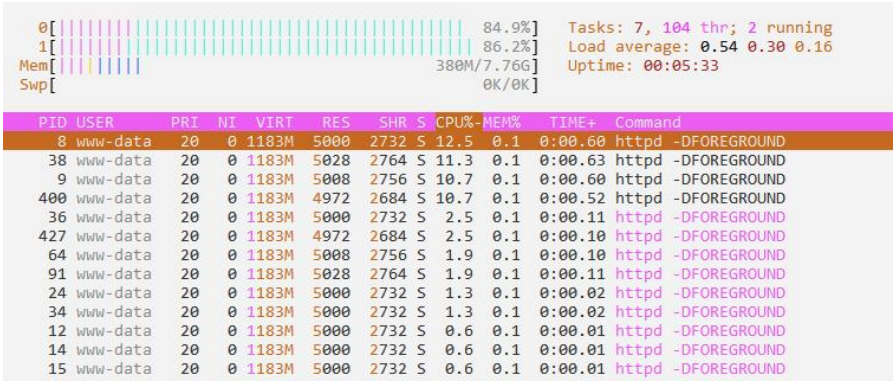
Pengambilan hasil data untuk skenario *intra-host* sama seperti pengambilan data pada komunikasi *inter-host*, hanya saja terdapat perbedaan pada alamat *web server Apache*. Alamat *web server Apache* yang diakses yaitu alamat IP *worker node2* yang terdapat pada *Google Cloud Platform*.

3.7.2 *Htop*

Pengambilan data CPU *usage* menggunakan aplikasi *htop*. Aplikasi *htop* berguna untuk melihat dan melakukan *monitoring* pengguaan kinerja *hardware*, sehingga dapat menentukan apakah *hardware* bekerja dengan optimal atau tidak. Aplikasi *htop* diinstal pada salah satu kontainer *worker node1* untuk komunikasi *inter-host* dan juga diinstal pada salah satu kontainer *worker node2* untuk komunikasi *intra-host*. Aplikasi *htop* diinstal menggunakan perintah dibawah.

```
root@7d26a62aff31:/usr/local/apache2# apt-get install htop
```

Pengambilan data dilakukan dengan memantau rata-rata CPU *usage* ketika dilakukan *benchmark* menggunakan aplikasi *Apache Benchmark*. *Htop* akan menampilkan data penggunaan CPU secara *realtime* seperti pada gambar 3.13.



Gambar 3.13 Pengambilan data CPU *usage* dengan *htop tools*