

Implementasi Load Balancing Dengan Algoritma Penjadwalan Weighted Round Robin Dalam Mengatasi Beban Webserver

Anggi Hanafiah and Rizky Wandri

Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Riau
anggihanafiah@eng.uir.ac.id, rizkywandri@eng.uir.ac.id

Article Info

History :

Dikirim 06 November 2020

Direvisi 14 November 2020

Diterima 06 Januari 2021

Kata Kunci :

Load Balance

Clustering

Linux Virtual Server

NAT

Weighted Round Robin

Abstrak

Dalam kehidupan sehari-hari semua orang tidak terlepas dari berbagai macam informasi, terutama informasi yang dihasilkan dari sebuah website. Selain dari pemrograman yang handal, resource yang lain seperti *webserver* juga sangat perlu diperhatikan agar website dapat berjalan dengan baik. Seiring meningkatnya kebutuhan konten dan pengunjung website, maka website sering mengalami crash atau request yang overload. Hal ini dikarenakan masih menerapkan *single server* untuk menangani website tersebut. Untuk mengatasi permasalahan tersebut, perlu diterapkan sebuah load balance cluster, dimana beban kerja *webserver* tersebut dapat didistribusikan ke beberapa node cluster. Hasil dari penelitian ini nantinya akan menghasilkan hasil perhitungan load balance dengan menggunakan *httpperf*, yang mana hasil yang dihitung antara penerapan *single server* dan load balance cluster. Algoritma penjadwalan *weighted round robin* merupakan salah algoritma penjadwalan dimana beban kerja *server* dapat berjalan seimbang dengan cara memberikan jumlah bobot ke masing-masing node cluster.

© This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Koresponden:

Anggi Hanafiah

Program Studi Teknik Informatika, Fakultas Teknik

Universitas Islam Riau,

Jl. Kaharuddin Nasution Perum. Dokagu Blok A.48, Pekanbaru, Indonesia, 28284

Email : anggihanafiah@eng.uir.ac.id

1. PENDAHULUAN

Perkembangan informasi pada saat ini sangatlah cepat. Hal ini bisa dilihat dari isi informasi yang disediakan sampai dengan bagaimana cara mendapatkan informasi tersebut. Dan dari cara mendapatkan informasi tersebut, informasi yang paling mudah dan sering diakses yaitu dengan media website.

Semakin berkembangnya website, baik dari sisi isi, tampilan maupun pengunjung yang semakin hari semakin meningkat, tidak sedikit website yang selalu mengalami *crash* maupun *overload* dalam pemrosesannya. Hal ini bukan hanya dilihat dari sisi pemrograman, melainkan *webserver* sebagai wadah penampung *website* tersebut sering mengalami kegagalan proses. Yang mana beban kerja dari *server* tersebut hanya dihandel oleh *single server*. [1]

Load balancing merupakan sebuah teknik untuk mendistribusikan trafik pada dua atau lebih jalur koneksi secara seimbang [2]. Salah satu algoritma penjadwalan dari *load balancing* yaitu

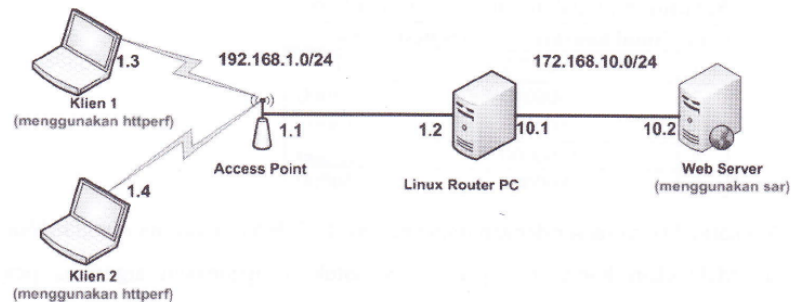
weighted round-robin. Algoritma *weighted round-robin* merupakan pengembangan dari algoritma *round-robin* yang dapat mempertimbangkan beban server berdasarkan kapasitasnya. Algoritma penjadwalan *weighted round-robin* memasukkan bobot atau parameter secara manual kepada masing-masing node *cluster* berdasarkan *resource*, sehingga *job scheduler* akan memprioritaskan job untuk server tersebut lebih dari server yang lainnya [3][4][5]. Dengan kata lain semakin besar *resource* yang dimiliki, semakin besar pula beban yang diberikan. Sehingga dengan adanya algoritma penjadwalan ini, selain dapat mengatasi beban dan meningkatkan ketersediaan yang tinggi, juga dapat meminimalkan *response time* dalam me-request website tersebut.

2. METODE PENELITIAN

Penelitian yang dilakukan yaitu perbandingan antara kinerja dari *webservice* yang dihandel hanya *single server* dan penerapan *load balance cluster* [6]–[8]. Paramater yang digunakan untuk memperlihatkan hasil perbandingan yaitu dari perhitungan terhadap *response time* yang diambil berdasarkan kecepatan *server*, serta pengujian *throughput* berdasarkan ketersediaan kecepatan jaringan dalam menyeimbangkan beban sistem *webservice*[9][10].

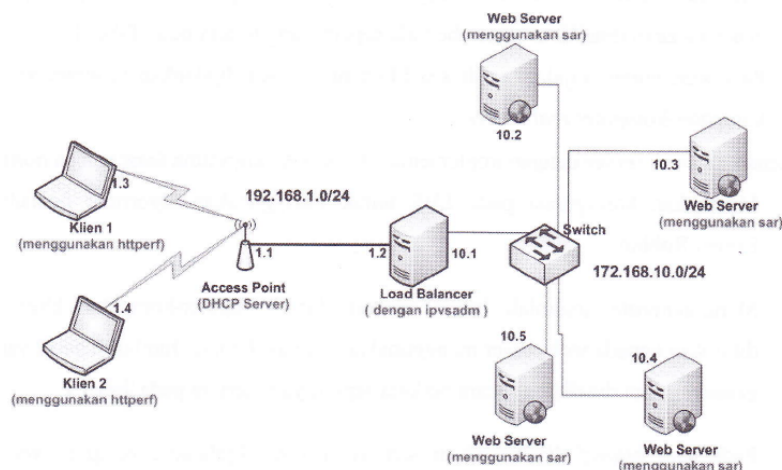
2.1 Rancangan Penelitian

Pada gambar 1 menunjukkan sebuah desain *webservice* dengan menggunakan *single server* sebagai penampung segala *request* yang diberikan oleh *client*.

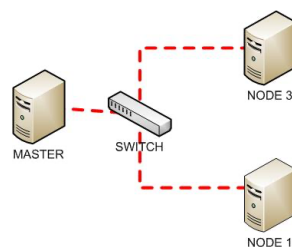


Gambar 1. Ilustrasi Topologi *Webserver* dengan *Single Server*

Pada gambar 2 menunjukkan bahwa segala beban *request* yang diberikan oleh *client* nantinya akan di terima oleh *server load balancer* sebagai *node master* sebelum nantinya akan diberikan ke masing-masing node *cluster* sesuai jumlah beban yang telah ditentukan



Gambar 2. Ilustrasi Topologi *Webserver* Dengan Penerapan *Load balance*



Gambar 3. Topologi *load balance cluster* dengan 3 node *cluster*

Pada pengimplementasian *load balance cluster* dalam penelitian menggunakan 1 buah *node master* sebagai *director* untuk melakukan proses kerja *webserver* biasa, dan 2 buah *node cluster* yang berfungsi membantu dalam mengatasi beban kerja dari *webserver* tersebut.

2.2 Linux Virtual Server

Linux Virtual Server yaitu proyek *open source* yang menyediakan *cluster server*, yang *high scalable* dan *high available* yang dibangun di atas sekumpulan beberapa *real server* dengan penyeimbang beban yang berjalan di sistem operasi Linux. Pada Linux Virtual Server terdapat 3 metode, yaitu Linux Virtual Server via NAT (*LVS-NAT*), Linux Virtual Server via Direct-Routing (*LVS-Direct Routing*), dan Linux Virtual Server via *Tunneling*.

2.3 Httperf

Httperf adalah *tool* yang dirancang untuk mengukur kinerja web server, yang memungkinkan administrator untuk mendiagnosis masalah potensial dan memperbaiki masalah yang mempengaruhi kinerja server mereka.

3. HASIL DAN PEMBAHASAN

3.1. Penerapan Load Balance Cluster

Penerapan *load balance* dimulai dengan menentukan spesifikasi serta penginstalan sistem operasi yang digunakan *server load balance* maupun *node cluster*. Pada penelitian ini, sistem operasi yang digunakan yaitu Ubuntu server versi 12.04 untuk *server load balance cluster* maupun *node cluster* yang akan digunakan.

Setelah penentuan spesifikasi, perlu dilakukan penginstalan serta pengkonfigurasiannya paket untuk *server load balance cluster* yaitu :

a. Ip Address

Dalam *server load balance* menggunakan 2 buah *ethernet card*, yang mana *ethernet* pertama (*eth0*) berfungsi sebagai penghubung ke masing-masing *node cluster*, sedangkan *ethernet* kedua (*eth1*) berfungsi sebagai *gateway* ke internet. Untuk pengkonfigurasiannya *IP Address* perlu diketikkan perintah berikut :

```
$ sudo su > untuk masuk ke super user
# nano /etc/network/interface > untuk masuk directory networking
# /etc/init.d/networking restart > untuk merestart networking
```

b. IP Virtual Server

Untuk memastikan sistem operasi mendukung service *IPVS*, maka perlu diketikkan perintah pada terminal sebagai berikut :

```
# grep -i ip_vs /boot/config-311.0-15-generic
```

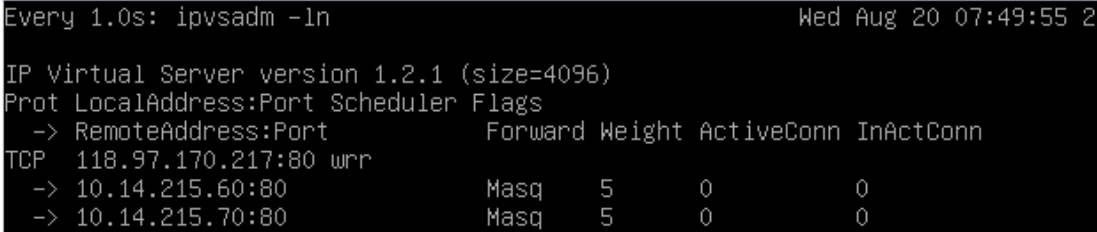
c. *Ipvsadm*

Didalam *ipvsadm* ini, perlu diperhatikan beberapa hal konfigurasi seperti pengaktifan *IP forwarding*, *NAT*, serta mengeset jumlah *node cluster* yang digunakan. Untuk melakukan set *ipvsadm* sekaligus menentukan jumlah *node cluster*, maka perlu diketikkan perintah berikut :

```
# ipvsadm -A -t 118.97.170.217:80 -s wrr
# ipvsadm -a -t 118.97.170.217:80 -r 10.14.215.60:80 -m -w 5 \\ config weighted round-robin
# ipvsadm -a -t 118.97.170.217:80 -r 10.14.215.70:80 -m -w 5 \\
```

Dan untuk konfigurasi *ipvsadm* agar tidak berubah dan memastikan status *ipvsadm* berjalan dengan baik serta, perlu diketikkan perintah berikut :

```
# ipvsadm -Sn
# watch -n1 ipvsadm -ln
```



```
Every 1.0s: ipvsadm -ln                               Wed Aug 20 07:49:55 2
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 118.97.170.217:80 wrr
-> 10.14.215.60:80          Masq   5       0         0
-> 10.14.215.70:80          Masq   5       0         0
```

Gambar 4. Tampilan Service *Ipvsadm*

Gambar diatas menunjukkan *Ipvsadm* dapat berjalan dengan menunjukkan jumlah *node cluster* sebanyak 2 buah *node cluster* dengan IP 10.14.215.60 dengan port 80, dan IP 10.14.215.70 dengan port 80 dengan beban yang seimbang.

d. *Ip forwarding*

Untuk mengaktifkan / enable *IP forwarding*, perlu diketikkan perintah berikut :

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
atau kita dapat mengubahnya dengan menggunakan perintah berikut :
# nano /etc/sysctl.conf
net.ipv4.ip_forward=1                                > Hilangkan tanda pagar
```

e. *NAT (Network Address Translation)*

Untuk mengaktifkan / enable NAT dengan enable ip masquerade, perlu diketikkan perintah berikut :

```
# iptables -t nat -A POSTROUTING -s 10.14.215.0/24 -o eth0 -j MASQUERADE
```

Penginstalan serta pengkonfigurasiannya yang dilakukan pada *node cluster* hampir sama dengan pada *server load balance cluster*, namun hanya saja paket yang dibutuhkan untuk *node cluster* ini hanya kebutuhan untuk *webservice* biasa.

3.2. Perbandingan antara *single server* dan *load balance cluster*

Pada pengujian yang dilakukan pada penelitian ini menggunakan aplikasi *benchmarking httpperf*, yang mana hasil dari pengujian *response time* yang diambil berdasarkan kecepatan *server*, serta pengujian *throughput* berdasarkan ketersediaan kecepatan jaringan dari segi *single server* maupun *load balance cluster*. Pengujian dengan *httpperf* dapat dijalankan dengan perintah berikut :

```
# httpperf --hog --server=118.97.170.217 --num-conns=1000 --ra=200 --timeout=10
```

```
root@webservers:/home/anggi# httpperf --hog --server=118.97.170.217 --num-conns=1000 --ra=200 --
timeout=10
httpperf --hog --timeout=10 --client=0/1 --server=118.97.170.217 --port=80 --uri=/ --rate=200 --send-
buffer=4096 --recv-buffer=16384 --num-conns=1000 --num-calls=1
httpperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to FD_SETSIZE
Maximum connect burst length: 1

Total: connections 1000 requests 500 replies 500|test-duration 21.094 s

Connection rate: 47.4 conn/s (21.1 ms/conn, <=1000 concurrent connections)
Connection time [ms]: min 4607.0 avg 9440.9 max 15407.7 median 10978.5 stddev 2638.3
Connection time [ms]: connect 3680.5
Connection length [replies/conn]: 1.000

Request rate: 23.8 req/s (41.9 ms/req)
Request size [B]: 67.0

Reply rate [replies/s]: min 0.0 avg 21.3 max 39.4 stddev 16.2 (4 samples)
Reply time [ms]: response 5155.6 transfer 0.0
Reply size [B]: header 359.0 content 251.0 footer 0.0 (total 610.0)
Reply status: 1xx=0 2xx=0 3xx=0 4xx=0 5xx=500

CPU time [s]: user 0.86 system 19.84 (user 4.1% system 94.1% total 98.1%)
Net I/O: 13.6 KB/s (0.1*10^6 bps)

Errors: total 575 client-timo 575 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
root@webservers:/home/anggi#
```

Gambar 5. Contoh Pengujian 200 Request Dengan *Single Server*

Pada tahapan hasil penelitian, percobaan dilakukan dengan melakukan pengujian dengan *single server* terlebih dahulu sebagaimana seperti gambar 5 diatas. Didalam pengujian ini dilakukan dengan memberikan koneksi sebanyak 1000 *request* dengan mengirimkan jumlah *request* secara bertahap sebanyak 200, 400, 600, 800 dan 1000 *request*/detik dari masing-masing model, baik dengan *single server* maupun dengan *load balance cluster*.

```

root@webserver:/home/anggi# httpperf --hog --server=118.97.170.217 --num-conns=1000 --ra=200 --
timeout=10
httpperf --hog --timeout=10 --client=0/1 --server=118.97.170.217 --port=80 --uri=/ --rate=200 --send-
buffer=4096 --recv-buffer=16384 --num-conns=1000 --num-calls=1
httpperf: warning: open file limit > FD_SETSIZE; limiting max. # of open files to FD_SETSIZE
Maximum connect burst length: 1

Total: connections 1000 requests 358 replies 358 test-duration 21.879 s

Connection rate: 45.7 conn/s (21.9 ms/conn, <=976 concurrent connections)
Connection time [ms]: min 819.8 avg 6090.8 max 16351.9 median 4133.5 stddev 4713.2
Connection time [ms]: connect 2452.8
Connection length [replies/conn]: 1.000

Request rate: 16.4 req/s (61.1 ms/req)
Request size [B]: 67.0

Reply rate [replies/s]: min 0.2 avg 2.2 max 4.8 stddev 2.0 (4 samples)
Reply time [ms]: response 3959.1 transfer 2131.7
Reply size [B]: header 284.0 content 30782.0 footer 2.0 (total 31068.0)
Reply status: 1xx=0 2xx=358 3xx=0 4xx=0 5xx=0

CPU time [s]: user 0.20 system 21.59 (user 0.9% system 98.7% total 99.6%)
Net I/O: 63.5 KB/s (0.5*10^6 bps)

Errors: total 955 client-timo 955 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
root@webserver:/home/anggi#

```

Gambar 6. Contoh pengujian *load balance cluster* dengan 200 *request/second*

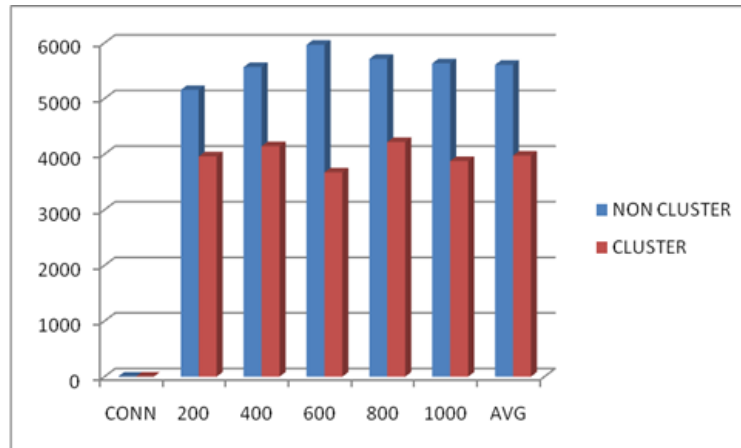
Gambar 6 diatas memperlihatkan bahwa pengujian *load balance cluster* dengan mengirimkan 200 *request* secara bersamaan dapat berjalan dengan baik. Didalam pengujian *load balance cluster* ini juga dilakukan dengan memberikan koneksi sebanyak 1000 *request* dengan mengirimkan jumlah *request* secara bertahap sebanyak 200, 400, 600, 800 dan 1000 *request/detik* dengan *load balance cluster*.

Hasil kinerja dari *load balance cluster* ini dapat dilihat dari pengujian *response time* yang diambil berdasarkan kecepatan *server*, serta pengujian *throughput* berdasarkan ketersediaan kecepatan jaringan sehingga dapat menyeimbangkan beban sistem *webserver*.

Tabel 1. Hasil perbandingan *response time* dengan *single server* dan *load balance LVS-NAT*

Total Connection	Single Server	Server Load Balance
200	5155,6	3595,1
400	5566,3	4142,7
600	5967,1	3672
800	5714	4219,7
1000	5632,5	3876,4
Rata-rata	5607,1	3876,4

Tabel 1 menjelaskan hasil dari perbandingan yang dilihat dari kecepatan *response time* terhadap *request* yang dilakukan secara bertahap dari 200 *request* bersamaan, hingga 1000 *request* secara bersamaan.



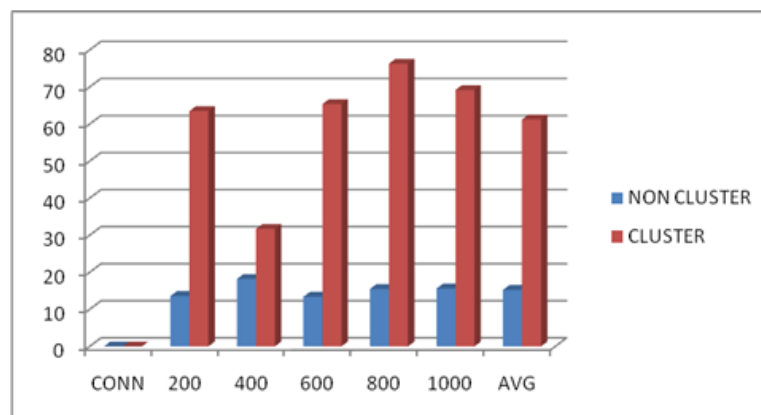
Gambar 7. Grafik perbandingan *Response time* dengan *single server* dan *load balance cluster*

Berdasarkan gambar 7 diatas, grafik menunjukkan hasil perbandingan yang mana perbandingan *response time* pada *single server* dengan penerapan *load balance cluster* menunjukkan dampak yang cukup signifikan.

Tabel 2. Hasil perbandingan *throughput* dengan *single server* dan *load balance LVS-NAT*

Total Connection	Single Server	Server Load Balance
200	13,6	63,5
400	18,2	31,7
600	13,4	65,4
800	15,5	76,3
1000	15,6	69,2
Rata-rata	15,26	61,22

Tabel 2 diatas menjelaskan perhitungan kecepatan *throughput* yang dihasilkan dari *single server* dan *load balance cluster* yang mana jumlah *request* yang diberikan sama dengan pengujian yang dilakukan dengan tabel 1 sebelumnya.



Gambar 8. Grafik perbandingan *throughput* dengan *single server* dan *load balance LVS-NAT*

Berdasarkan gambar 8 diatas, grafik menunjukkan hasil yang sangat signifikan terhadap kecepatan *throughput* dengan menggunakan *load balance cluster* dibandingkan dengan penerapan *single server*.



4. KESIMPULAN

Kesimpulan dalam pengujian penelitian ini yaitu Hasil kinerja *load balance cluster* ini dapat dilihat dari 2 hal yaitu kecepatan *response time* dan *throughput*. Dan Hasil kinerja *load balance cluster* dilihat dari *response time* mempunyai dampak yang cukup signifikan, sedangkan dilihat dari *throughput* mempunyai dampak yang sangat signifikan. Untuk mendapatkan hasil yang lebih signifikan, maka diperlukan beberapa *node cluster* pendukung minimal dengan 10 node atau lebih.

DAFTAR PUSTAKA

- [1] A. Rahmatulloh and M. S. N. Firmansyah, "Implementasi load balancing web server menggunakan haproxy dan sinkronisasi file pada sistem informasi akademik Universitas Siliwangi," *J. Nas. Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 241–248, 2017.
- [2] D. K. Hakim, J. K. Riyanto, and A. Fauzan, "Pengujian Algoritma Load Balancing pada Virtualisasi Server," *Sainteks*, vol. 16, no. 1, 2020.
- [3] C. EL AMRANI and H. GIBET TANI, "Smarter round robin scheduling algorithm for cloud computing and big data," *J. Data Min. Digit. Humanit.*, 2018.
- [4] D. Biswas and M. Samsuddoha, "Determining Proficient Time Quantum to Improve the Performance of Round Robin Scheduling Algorithm," *Int. J. Mod. Educ. Comput. Sci.*, vol. 11, no. 10, p. 33, 2019.
- [5] Y. Arta, "Penerapan Metode Round Robin Pada Jaringan Multihoming Di Computer Cluster," *Inf. Technol. J. Res. Dev.*, vol. 1, no. 2, pp. 26–35, 2017.
- [6] H. Wang, Y. Yang, B. Liu, and H. Fujita, "A study of graph-based system for multi-view clustering," *Knowledge-Based Syst.*, vol. 163, pp. 1009–1019, 2019.
- [7] Y. Arta, "Implementasi Computer Cluster Berbasis Open Source Untuk menyeimbang Beban Sistem Dan Jaringan Komputer," *J. Tek. Inform. dan Sist. Inf.*, vol. 2, no. 1, 2016.
- [8] V. Y. Kiselev, T. S. Andrews, and M. Hemberg, "Challenges in unsupervised clustering of single-cell RNA-seq data," *Nat. Rev. Genet.*, vol. 20, no. 5, pp. 273–282, 2019.
- [9] Y. Liu, M. Grimm, W. Dai, M. Hou, Z.-X. Xiao, and Y. Cao, "CB-Dock: a web server for cavity detection-guided protein–ligand blind docking," *Acta Pharmacol. Sin.*, vol. 41, no. 1, pp. 138–144, 2020.
- [10] F. Ahmed *et al.*, "pssRNAit: A Web Server for Designing Effective and Specific Plant siRNAs with Genome-Wide Off-Target Assessment," *Plant Physiol.*, vol. 184, no. 1, pp. 65–81, 2020.

BIOGRAFI PENULIS

	<p>Anggi Hanafiah is a Lecturer of Department of Informatics Engineering, Islamic University of Riau. Obtained Bachelor Degree in Informatics Engineering from STMIK-AMIK Riau , obtained Master Degree in Information Engineering from UPI-YPTK Padang. His current research interests include Data Mining, Artificial Inteligent, Networking, and Multiplatform Programming.</p>
	<p>Rizky Wandri is a Lecturer of Department of Informatics Engineering, Islamic University of Riau. Obtained Bachelor Degree in Informatics Engineering from STMIK Hangtuah , obtained Master Degree in Information Engineering from UPI-YPTK Padang. His current research interests include Web Programing and Data Mining.</p>