BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Penelitian Pertama dilakukan oleh (Mandahadi Kusuma, dkk 2016) yang berjudul *"*Evaluasi Performa Web Server Menggunakan Varnish HTTP Reserve Proxy dan Redis Database Cache". Hasil Penelitian menunjukkan bahwa penggunaan varnish dan redis-cache pada webserver dengan total 20000 hit/request alamat web, error rate dapat diturunkan dari 10,38% hingga menjadi 2,16% (~3 kali lebih rendah), response time menjadi 15-16 kali lebih cepat dibandingkan webserver yang tidak menggunakan cache.

Penelitian kedua oleh (Anindya Putri Arunawati/ 2020) dengan judul "Optimasi Apache Web Server Menggunakan Varnish Web Cache Dan Reverse Proxy Nginx*"*

Penelitian menghasilkan bahwa apache web server yang telah dioptimasi menunjukkan performa lebih baik dibandingkan sebelum dioptimasi atau tanpa reverse proxy dan web cache.

Penelitian Ketiga (Luthfi *et al*., /2018) dengan judul "Perbandingan Performa *Reverse Proxy Caching Nginx* dan *Varnish Pada Web Server Apache"* Penelitian menghasilkan bahwa s*erver apache* dengan *reverse proxy caching varnish* memiliki performa yang lebih baik dibandingkan dengan *server apache* dengan *reverse proxycaching Nginx*.

Penelitian keempat oleh (A.Chandra /2019) dengan judul "Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request" dengan hasil penelitian bahwa benchmarking menunjukkan bahwa dari sisi penggunaan waktu, nginx menggunakan waktu lebih sedikit daripada Apache dalam menyelesaikan client request

Penelitian kelima oleh (Muhammad Syaifuladnan, 2017) dengan judul Analisis Kinerja Web server Dengan Metode *Load Balancing* Pada Haproxy . Saat ini perkembangan internet sangat pesat sekali, seiring dengan semakin banyaknya user yang terhubung ke jaringan internet. Tapi masih banyak yang menggunakan server tunggal dan mendapatkan request dari banyak user, dalam hal ini dapat memungkinkan terjadinya overload dan crash sehingga request dari banyak user tidak dapat di layani dengan baik oleh server tunggal.

Tabel 2.1 Tinjauan Pustaka

No	Judul	Oleh/tahun	Objek	Metode	Hasil
1	Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request	A.Chandra /2019 Mandahadi Kusuma, dkk 2016	*web* server Apache	Bench Marking	benchmarking yang menunjukkan bahwa dari sisi penggunaan waktu, nginx menggunakan waktu lebih sedikit daripada Apache dalam menyelesaikan client request
2	Optimasi Apache Web Server Menggunakan Varnish Web Cache Dan Reverse Proxy Nginx	Anindya Putri Arunawati/ 2020	*web* server	Metode : • Varnish • Nginx	apache web server yang telah dioptimasi menunjukkan performa lebih baik dibandingkan sebelum dioptimasi atau tanpa reverse proxy dan web cache
3	Perbandingan Performa *Reverse Proxy Caching Nginx* dan *Varnish Pada Web Server Apache*	Luthfi *et al*., /2018 .	*web* server Apache	perbandingan	Hasil *server apache* dengan *reverse proxy caching varnish* memiliki performa yang lebih baik dibandingkan dengan *server apache* dengan *reverse proxycaching Nginx*.

4	Evaluasi Performa Web Server Menggunakan Varnish HTTP Reserve Proxy dan Redis Database Cache		*web* server Apache	Varnish web cahe	dengan total 20000 hit/request alamat web, error rate dapat diturunkan dari 10,38% hingga menjadi 2,16% (~3 kali lebih rendah), response time menjadi 15-16 kali lebih cepat dibandingkan webserver yang tidak menggunakan cache.
5	Analisis Kinerja Web server Dengan Metode *Load Balancing* Pada Haproxy	Muhammad Syaifuladnan / 2017	*Web server Haproxy*	Load Balancing	Hasil yang diperoleh dari multi node dapat digunakan untuk mengembangkan metode load balancing.

2.2. Landasan Teori

2.2.1. Jaringan Komputer

Pada era globalisasi zaman sekarang, kemajuan dan perkembangan dunia teknologi dan internet sangatlah pesat sehingga berdampak di setiap aktivitas perusahaan, instansi atau lainnya, sebagai kebutuhan pokok dalam media komunikasi antar cabang perusahaan kantor pusat atau sebagai media dalam pertukaran data (*sharing*) antar cabang yang dapat diakses melalui jaringan komputer (Akbar *et al*, 2019: 85). Jaringan komputer (*computer networks)*

adalah suatu himpunan interkoneksi sejumlah komputer *autonomous.* Jaringan komputer merupakan kumpulan beberapa komputer dan perangkat jaringan komputer seperti*router, switch, hub,* dan sebagainya (Sofana, 2015:3).

2.2.2. Server

Server merupakan sebuah komputer dengan prosessor yang mempunyai inti lebih dari satu yang dapat menjalankan aplikasi aplikasi dan services secara bersamaan (Putra, 2019: 2). *Server* secara sederhana dapat berupa satu buah komputer untuk beberapa layanan aplikasi, atau jika jaringannya lebih komplek dan rumit, maka *server* dapat diatur hanya untuk memberikan satu atau beberapa layanan saja, sementara layanan yang lain diserahkan kepada *server* yang lain. Untuk *server* yang terdiri dari satu buah komputer yang melayani beberapa layanan biasanya hanya digunakan untuk lingkungan yang lebih kecil misal sekolah, perkantoran, atau usaha kecil dan menengan (UKM) (Suryana, 2018: 1). Berdasarkan fungsinya *server* dapat dibedakan menjadi :

1. *Web server* : *server* yang berfungsi untuk memberikan layanan protokol http, contoh aplikasi *web server* yaitu : *apache, Microsoft IIS, Tomcat, Nginx,* dll
2. *Database server : server* yang berfungsi untuk menyimpan data secara terpusat dan mendistribusikan ke *client* melalui jaringan *wireless* ataupun kabel, Contoh *database server : MySQL, Postgres, MS SQL Server, Oracle, Interbase,* dll
3. FTP *Server* : *Filezilla, FTPd, pro-FTPd, Wu-FTPd, ftpX , Troll-FTPd.*

4. *Mail Server : Mercury, Merak, sendmail, postix.*
5. *Print / File server : Samba Serve*
6. DNS *Server : Server* yang berfungsi menerjemahkan alamat *host* menjadi IP

   address, contoh : Bind.
7. DHCP *Server* : *server* yang bertugas memberikan IP *address* secara otomatis kekomputer *client.*
8. *Proxy server* : aplikasi ini diterapkan untuk membatasi hak akes ke internet ataupun ke suatu *server*. sehingga dapat dibatasi jumlah pengguna ataupun adanya saringan ke media masa, mana saja yang dapat diakses.

Semuanya memiliki fungsi yang berbeda dan tanpa mereka tidak mungkin mempertahankan jaringan (Rahman, 2018: iv).

2.2.3. Web Server

Web server merupakan suatu perangkat lunak yang berjalan di sisi *server* dan bertugas untuk menerima permintaan dari *web browser*, menerjemahkan permintaan tersebut, dan mengembalikan ke *web browser* hasil dari permintaan itu Dalam bentuk sederhana *web server* akan mengirim data HTML kepada permintaan *web browser* sehingga akan terlihat seperti pada umumnya yaitu sebuah tampilan *website*. Fungsi utama *web server* adalah untuk melakukan atau akan *tranfer* berkas permintaan pengguna melalui protokol komunikasi yang telah ditentukan sedemikian rupa. Pemanfaatan *web server* berfungsi untuk men-*transfer* seluruh aspek pemberkasan dalam sebuah halaman *web* termasuk yang

di dalam berupa teks, video, gambar atau banyak lagi. *Web* dapat dibangun dengan menggunakan bahasa HTML dan PHP dengan *style* tampilan menggunakan bahasa CSS. *Web* tersebut disimpan pada satu komputer yang disebut *server* (Dawood *et al,* 2014: 26).

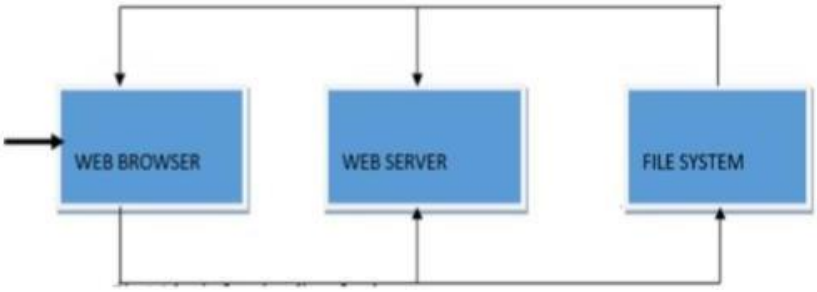
Hubungan antara *web server* dan *browser internet* merupakan gabungan atau jaringan komputer yang ada di seluruh dunia. Setelah terhubung secara fisik, *Protocol* TCP/IP (*networking protocol*) memungkinkan semua komputer dapat berkomunikasi satu dengan yg lainnya. Pada saat *browser* meminta data *web page* ke *server* maka instruksi permintaan data oleh *browser* tersebut di kemas di dalam TCP yg merupakan *protocol transport* dan dikirim ke alamat menggunakan *protocol* berikutnya yaitu *Hyper Text Transfer Protocol* (HTTP). Data yg di passing dari *browser* ke *web server* disebut sebagai HTTP *request*. Data yg dikirim dari *server* ke *browser* disebut sebagai HTTP *response*. Jika data yg diminta oleh *browser* tidak ditemukan oleh *web server* maka akan menampilkan *error* yang sering anda lihat di *web* page yaitu e*rror* : 404 *Page Not Found* (Yogiswara, 2015:176).

Beberapa jenis *webserver* di antanya adalah:

1. *Apache Web Server / The HTTP Web Server*
2. *Apache Tomcat*
3. *Microsoft windows Server 2008 IIS (Internet Information Services)*
4. *Lighttpd*
5. *Zeus Web Server*
6. *Sun Java System Web Server*

7. Nginx

Fungsi utama dari *web server* adalah untuk melayani permintaan dari *client*. Sebuah kontak *client web server* melalui jaringan, mengirimkan permintaan yang berisi nama *file* yang diperlukan dan kemudian *server* merespon dengan isi *file* jika ada, operasi dasar dari proses ini dapat digambarkan dari Gambar 2.1. (Kunda *et al*, 2017: 43).



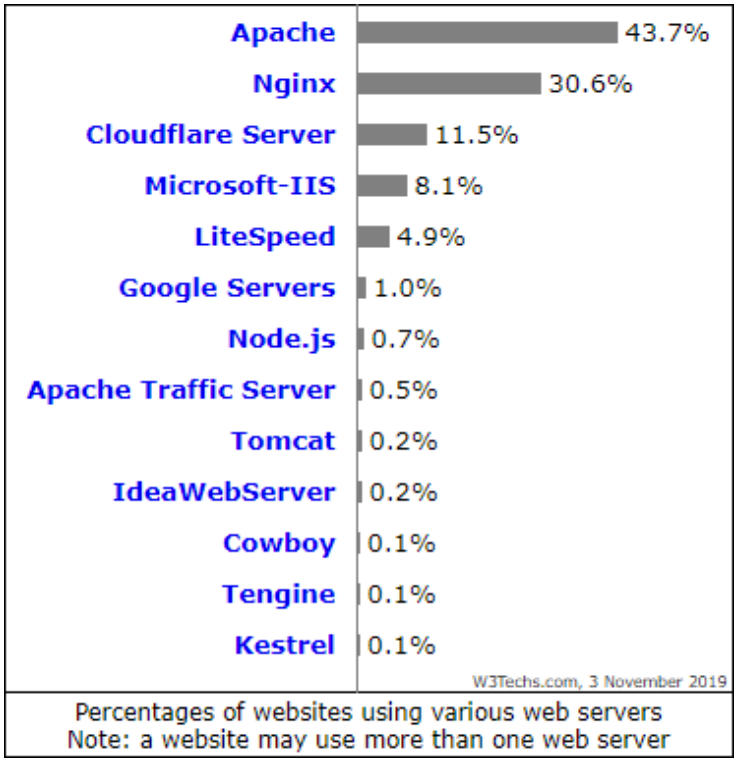
Gambar 2.1 Dasar Fungsi *Web Server*

Saat ini, sebagian besar aplikasi berbasis *web*. *Web server* memainkan peran penting dimana *web server* menangani semua permintaan pengguna. Perlambatan di tingkat *web server* akan mempengaruhi pengalaman pengguna. Sebagai kinerja aplikasi *web* semakin semakin penting, karena jumlah pengguna dan pesaing masih meningkat, untuk alasan ini, penelitian terbaru telah secara komprehensif dieksplorasi metode untuk mengevaluasi atau mengoptimalkan kinerja *web server* (Xu *et al*, 2013: 1). Kinerja *web server* mengacu pada kemampuan dari *web server* menanggapi permintaan, yang secara langsung mempengaruhi kepuasan pengguna (Qu *et al*, 2010: 1).

2.2.4. Apache

Menurut lembaga *survey* Netcraft (2017) *Apache HTTP Server* adalah perangkat lunak *web server open-source*. *Open-source* yang dikembangkan dan dikelola oleh *Apache Software Foundation*. Rilis pada tahun 1995, *apache* menjadi perangkat lunak *web server* yang paling banyak digunakan hingga saat ini. Sebuah survei yang dilakukan oleh Netcraft menunjukkan bahwa pada Februari 2017*Apache* digunakan di 45,78% situs aktif .

Berdasarkan *survey* dari lembaga *survey* W3Tech *apache web server* juga menempati posisi teratas dengan pengguna sebanyak 43, 7% dan pada urutan kedua terdapat *nginx web server* dengan pengguna sebanyak 30,6% seperti pada Gambar 2.2. Ini membuktikan bahwa sampai saat ini *apache web server* masih menjadi *webserver* yang digemari dan banyak digunakan.



Gambar 2.2 *Survey Web Server*

Beberapa dukungan yang terdapat pada *apache web server* yang menjadialasan *client* untuk menggunakannya dibanding *web server* yang lain diantaranya :Kontrol akses. Kontrol ini dapat dijalankan berdasarkan nama *host*.

1. *CGI (Common Gateway Interface)* Yang paling terkenal untuk digunakan adalah *perl* (*Practical Extraction and Report Language*), didukung oleh *apache* denganmenempatkannya sebagai modul (*mod_perl*)
2. PHP (*Personal Home Page/PHP Hypertext Processor*) Program dengan metode semacam CGI, yang memproses teks dan bekerja di *server*. *Apache* mendukung PHP dengan menempatkannya sebagai salah satu modulnya (mod_php).

3. SSI (*Server Side Includes*)

Keunggulan *apache* dibandingkan dengan *web server* yang lain seperti IIS (*Internet Information Service*) dari *microsoft* adalah kemampuannya untuk mendukung berbagai bahasa *script* paling populer seperti PHP (*Personal Home Page*) dan JSP (*Java Server Pages*). Hal lain yang membuat *apache* lebih diminati adalah sistem lisensinya yang gratis sehingga mengurangi biaya yang perlu dikeluarkan dalam membangun situs *web* dinamis (Emanuel, 2014: 23)

Apache adalah salah satu *web server* yang bertanggung jawab pada *request- response HTTP* dan *logging* informasi secara detail. Secara umum arsitektur apache bekerja dengan model thread-based atau berbasis proses. *Web server* berbasis proses menggunakan proses (*thread*) untuk menerima dan merespon permintaan. Setiap permintaan akan diciptakan sebuah thread yang disimpan dalam sebuah *pool* pada alokasi memori tertentu dan dilakukan proses untuk menjawab, setelah proses dieksekusi kemudian hasil proses dikirimkan kembali ke klien serta dicatat dalam sebuah *log* (Yogiswara, 2015: 176).

Selain itu, a*pache* juga terkenal sebagai suatu *web server* yang kompak, modular, mengikuti standar protokol HTTP, dan tentu saja sangat digemari. Kesimpulan ini bisa didapatkan dari jumlah pengguna yang jauh melebihi para pesaingnya (Irza *et al*, 2017: 76).

Fungsi dari *handler* melakukan apa yang perlu dilakukan untuk memenuhi permintaan, kemudian mengirimkan proses itu kembali ke komponen HTTP_REQUEST dari inti *apache* agar dikirim ke modul lain untuk diproses ataudikembalikan ke *client* (Irza *et al*, 2017: 76).

2.2.5. Varnish Web Cache

Untuk meningkatkan kecepatan akses pada aplikasi berbasis *web* digunakan *cache*. *Web cache* berada diantara *web server* dan *client*. Ketika ada *request* dari *client*, *cache* akan menyimpan respons, dalam bentuk html, gambar, atau *file*. Selanjutnya apabila ada *request* yang sama dari sisi *client*, maka cache server akan langsung memberikan respons, tidak perlu kembali mengakses *web server* asli. Tujuan utama *web cache* untuk mengurangi *latency* (waktu tunda) dan mengurangi beban network server utama. Secara umum ada 3 jenis *web cache*, yaitu; *browser cache*, *proxy cache*, dan *gateway cache* (Kusuma *et al*, 2016: 260)

Cache HTTP *server* (akselerator *web*) biasanya digunakan antara *client* dan *server* HTTP dengan melayani *client* dengan mengirimkan *file* yang disimpan dalam *cache sever* HTTP *(web server*). *Varnish* secara fungsi hampir sama dengan *nginx-cache*, perbedaannya ada pada hal penyimpanan *cache* nya. Jika pada *nginx*, *cache* disimpan ke *disk* di *server*, berbeda dengan *varnish* yang menyimpan *cache* nya di memori. Banyak yang *meng-claim* bahwa *caching* menggunakan *varnish* akan lebih cepat karena *cache* yang disimpan di memori tersebut. Namun karena memang memori memiliki limitasi dari pada *disk* dalam hal ukuran, maka *varnish* harus dikonfigurasi dengan tepat (Xie *et al*, 2012: 1).

Varnish adalah perangkat lunak gratis yang dilisensikan dengan lisensi BSD dua klausa, juga dikenal sebagai lisensi *FreeBSD*. Proyek ini dimulai pada tahun 2005. *Versi* pertamanya adalah dirilis pada September 2006 (*Varnish cache* 1.0), dan versi terbaru *Varnish cache* 3.0.2 dirilis pada Oktober 2011. Fitur

utama *Varnish* adalah miliknya kinerja dan fleksibilitas. Melalui bahasa konfigurasi sendiri vcl (*Varnish Configuration Language*) sangat dapat dikonfigurasi, dan pengguna dapat membuat kebijakan bagaimana menangani berbagai skenario lalu lintas (Bakhtiyari, 2012: 27).

Varnish cache adalah sebuah aplikasi *Web accelerator* yang dikenal juga sebagai *Hyper Text Transfer Protocol* (HTTP) *reverse proxy*. *Varnish* diperkenalkan pertama kali pada tahun 2006 oleh *Poul-Henning Kamp*. *Reverse proxy* adalah sebuah *proxy* yang berada di *front end web server* berfungsi sebagai *cache*. *Varnish* bekerja pada *front end* dari sebuah *server HTTP* dan dapat mengkonfigurasi *cache* tiap konten *website*. Prinsip kerja utama *Varnish* adalah menyimpan data halaman *website* di dalam *memory*, sehingga mengurangi beban *web server* memuat halaman yang sama (Kusumo *et al*, 2014: 2).

Varnish Configuration Languange (VCL) merupakan bahasa pemrograman yang digunakan dalam *varnish web cache*. *Syntax* pemprograman pada VCL mirip dengan pemrograman C. Proses *request varnish* dibagi dalam tiga bagian utama yaitu *vcl_recv, vcl_fetch,* dan *vcl_delive.* Varnish mempunyai dua konfigurasi yang berbeda yaitu satu ada di file */etc/varnish/varnish.params* dan satu lagi ada di */etc/varnish/defaul.vcl* (Kusumo *et al*, 2014:2).VPS (Virtual Private Server)

Virtualisasi *server* menjadi teknologi yang saat ini tumbuh dengan cepat. Terbukti berdasarkan laporan survei diterbitkan oleh spiceworks.com, menyatakan bahwa 76% atau lebih dari tiga perempat responden menggunakan

virtualisasi *server* di internet (Soni *et al*, 2019: 18). *Virtual Private Server* lebih dikenal dengan singkatan VPS. Ini adalah sebuah metode untuk mempartisi sebuah komputer *server* menjadi beberapa *server* yang sepertinya *server-server* tersebut berdisi sendiri, seolah-olah sebagai *server* mandiri dan berlaku seperti layaknya sebuah komputer. Meskipun kenyataannya dalam satu komputer terdapat beberapa akun VPS, tetapi satu akun dengan akun lain tidak akan saling mempengaruhi. Hal ini dapat membantu mengimprovisasi tingkat fleksibilitas yang tersedia pada administrator sistem dalam hal pemilihan konfigurasi *software* yang dapat dijalankan. Selain itu, ini juga dapat memberikan keuntungan yang signifikan dalam hal skalabilitas dari daya pemrosesan (*processing power*), RAM, dan *disk space* dengan biaya yang lebih rendah daripada menggunakan *hardware fisik* tradisional. (Alamsyah & SmitDev, 2009: 31).

Virtualisasi *server* adalah teknik yang digunakan untuk mengkonfigurasi dan mengoperasikan sejumlah *server virtual* dengan membagi sumber daya dari *server* fisik. Ini memberikan sejumlah keuntungan termasuk manajemen sumber daya yang efisien, menghemat ruang, memperkuat keamanan, dan penghematan biaya (Jung *et al*, 2011: 40).

Server virtual secara logis bekerja secara terpisah meskipun secara fisik terletak di sama perangkat keras. Ketika beberapa *server virtual* dibuat dan memiliki beban kerja yang berat, *server* dengan sumber daya terbatas harus mampu mengelola penggunaan sumber daya di antara *vitual privat server. vitual privat* server (VPS) hosting adalah salah satu layanan *hosting* yang paling

banyak digunakan untuk *website*. Tipe *hosting* ini menggunakan teknologi virtualisasi yang menyediakan *resourse dedicated* (pribadi) di *server* meskipun digunakan oleh lebihdari satu *user* (Sabri & Musa, 2013: 1).

2.2.6. Apache Benchmark

ApacheBench adalah alat untuk proses *benchmark apache* HTTP *server* dan di desain untuk memberikan gambaran performa instalasi *apache*. Secara khusus akan menampilkan seberapa banyak *request* per detik yang bisa dilayani oleh *apache* (Kusumo *et al,* 2014:2). Tujuan *benchmark* adalah untuk mengevaluasi dan menggambar perbandingan antara kerangka kerja komputasi aliran terdistribusi yang berbeda. Selain pengukuran kinerja, tolok ukur juga harus memperhatikan sifat yang harus didistribusikan dari kerangka kerja target, oleh karena itu dapat mempertimbangkan kemampuan toleransi kesalahan, ketersediaan, dan skalabilitas(Lu *et al,* 2014: 70).

Apache benchmark adalah *tools* yang digunakan untuk mengukur kinerja *server web HTTP*. *Apache benchmark* awalnya dirancang untuk menguji s*erver HTTP Apache,* tetapi *apache benchmark* ini bisa digunakan tidak hanya untuk *apache* bisa juga untuk *web server* lain juga seperti *lighthttpd, nginx* atau *miscrosoft iis*. *Apachebench* dengan total koneksi N dibuat dengan konkurensi koneksi C, di mana N dan C ditentukan oleh pengguna. Utilitas *ApacheBench* mencatat *respons, throughput*, dan statistik *server* lainnya selama pengujian. Pada dasarnya, masalah utilitas *apachebench HTTP GET* permintaan untuk konten dalam menetapkan tolok ukur kinerja dari *web server*. Dari hasil pengujian

menggunakan *apache benchmark* yang harus diperhatikan adalah berapa hasil dari *requests per second* karena ini menunjukkan seberapa banyak pengunjung yang dilayani dalam satu detik, tapi ingat satu pengunjung bisa membuka beberapa koneksi/permintaan konten sekaligus, dan angka pada *Percentage of the requests served within a certain time (ms),* ini menandakan seluruh permintaan koneksi berhasil dilayankan dalam waktu berapa milidetik. Bagi 1000 untuk mendapatkan 1 detik, jadi kalau dari hasil benchmark diatas paling lama bisa mencapai 3.2 detik. Ini kalau anda matikan parameter header GZIP dan *keep alive* bisa melonjak ke 10 detik lebih (Brunelle *etall*, 2013: 5).

2.2.7. Optimasi

Optimasi adalah proses pencarian satu atau lebih penyelesaian yang berhubungan dengan nilai-nilai dari satu atau lebih fungsi objektif pada suatu masalah sehingga diperoleh satu nilai optimal.1 Secara umum optimasi berarti pencarian nilai terbaik (minimum atau maksimum) dari beberapa fungsi yang diberikan pada suatu konteks. Optimasi juga dapat berarti upaya untuk meningkatkan kinerja sehingga mempunyai kualitas yang baik dan hasil kerja yang tinggi.