# BAB II

# Tinjauan Pustaka dan Landasan Teori

## 2.1 Tinjauan Pustaka

Studi atau penelitian mengenai *load balance* sudah terbilang cukup banyak dilakukan dan diimplementasikan. Beberapa diantaranya penelitan oleh Alam Rahmatulloh, dan Firmansyah MSN [1], Deepti Sharma dan Vijay B. Aggarwal [2], Nanang Basir Umkabu, Bayu Erfianto, Endro Ariyanto [3], dan Ari Budi Noviyanto, Erna Kumalasari, serta Amir Hamzah [4]. Keempat penelitian tersebut menerapkan *load balance server* menggunakan Apache *http web server* sebagai *backend server*. Namun demikian Apache memiliki kelemahan [5], diantaranya:

1. Kurang handal dalam menangani jumlah *user request* yang besar.
2. Kurang handal dalam mengelola *static content.*
3. Terbatasnya fitur dan fungsi *limitation* berdasarkan *ip address*.
4. Konsumsi *bandwith*, CPU, serta alokasi *memory server* yang terbilang boros.
5. Dalam menangani jumlah *request*, Apache membutuhkan waktu yang lebih lama dibandingkan Nginx, seperti terlihat pada tabel 2.1 [2].

**Tabel 2.1** Nginx dan Apache Jumlah Proses berdasarkan Waktu [2]

| *No. of Requests* | *Total Processing Time (sec)* | |
|---|---|---|
| | *Apache Load Balancer* | *Nginx Load Balancer* |
| 100 | 14.7 | 8.39 |
| 400 | 22.8 | 18.05 |
| 900 | 52.2 | 36.6 |
| 1600 | 177.2 | 59.5 |
| 2500 | 192.69 | 124.31 |
| 6400 | 292.52 | 186.78 |
| 10000 | 538.55 | 346.09 |

Sedangkan Nginx dibandingkan dengan Apache memiliki beberapa keunggulan [6, 7] diantaranya:

1. Handal dalam menerima jumlah *request client* yang besar.
2. Cepat dan *convergent* dalam memproses *request* dan respon.
3. Multi fungsi, dapat dioperasikan sebagai *web server*, *proxy server*, *load balancer*, *caching content*, dan lainnya.
4. Konsumsi *bandwith* serta *memory server* lebih sedikit dibandingkan dengan Apache.
5. Memiliki fitur DOS/DDOS proteksi.
6. Konfigurasi yang fleksibel mengikuti kehendak dari pengguna.

Selain itu, Matthew Prince, *co-founder* sekaligus *CEO* dari *CloudFlare* [7] menyatakan bahwa, Nginx sebagai *web server* lebih unggul dibandingkan dengan Apache, dikarenakan adanya fitur pembagian dan pembatasan alokasi *memory buffer.* Aplikasi pada Nginx berjalan sesuai dengan kebutuhan *memory* per *server*, dibuktikan dengan *CloudFlare* yang memiliki *request* lebih dari 15 juta *request* per bulannya pada *website* mereka yang menggunakan Nginx dan masih dapat berjalan dengan baik.

Berdasarkan dari perbandingan serta studi atau kajian yang telah disebutkan, maka dalam penelitian kali ini penulis memilih menggunakan Nginx sebagai *load balance server* dan *backend server*.

Selain menggunakan Apache, dari keempat penelitian peneliti sebelumnya yang telah disebutkan, juga menggunakan HAProxy sebagai aplikasi tambahan yang berfungsi sebagai *load balancer*. Penggunaan HAProxy dalam *Application Delivery Controller* (*Load Balance*) tidak begitu direkomendasikan [8], terlebih pada penelitian

ini, Nginx telah memiliki fungsi sebagai *load balancer server*. Selain itu, seperti terlihat pada gambar 2.1, HAProxy tidak terdapat pada *Market leaders* atau *challengers* untuk *Application Delivery Controller* (*Load Balance*) [8]. Sedangkan Nginx masuk ke dalam kategori *Niche Player* yang artinya menjadi pilihan *alternative* selain *Challengers* dan *Leaders*.


**Gambar 2.1** Gartner *Magic quadrant Application Delivery Controller.* Diadopsi dari Gartnet Quadrant [8]

Di sisi lain, terdapat penelitian sebelumnya terkait dengan *load balance* yang dilakukan oleh Iwan Rijayana [9]. Dalam penelitiannya Iwan Rijayana memberikan gambaran mengenai proses *load balancing*, serta *software* dan *hardware* yang dibutuhkan. Dalam paparannya ia menggunakan *Windows 2000 Advanced Server Service Pack 2* sebagai OS dan *Microsoft Exchange 2000 ASP* sebagai *web server* yang menjalankan *load balancing*. Namun *Operating System* (OS) dan *web server* tersebut merupakan aplikasi atau *software* berlisensi yang memiliki harga mahal. Selain itu keduanya merupakan *software* atau aplikasi *close sources*, yang artinya tidak terdapat

kebebasan pada pengguna terhadap *operating system* tersebut untuk memodifikasi sesuai dengan kehendak pengguna. Pada penelitian kali ini penulis menggunakan Ubuntu 14.04 LTS yang merupakan *operating system free* dan *open source* sebagai alat atau *tools* yang menjalankan layanan Nginx *load balance*. O*perating system free* dan *open source* ini lebih cocok untuk organisasi atau perusahaan yang memiliki dana terbatas.

Di sisi lain terdapat pula penelitian yang dilakukan oleh Supraman dan I Gusti Lanang Putra Prismana [10]. Supramana dan I Gusti Lanang Putra Eka Prismana pada penelitiannya, melakukan penelitian menerapkan *http web server load balance* menggunakan Apache sebagai *web server* yang mendukung di dalam penelitiannya dengan metode *reverse proxy* sebagai *load balancer server*, lalu dijalankan pada emulator *software* yaitu GNS3. Sedangkan pada penelitian ini, pada *device* penulis menggunakan *server* asli (*real device*) yang divirtualisasikan menggunakan proxmox. Menggunakan *real device* memiliki keuntungan diantaranya mampu menggambarkan keadaan yang sebenarnya dilapangan serta dapat merasakan konfigurasi pada perangkat secara langsung.

Berdasarkan hasil tinjauan pustaka maka didapatkanlah hasil serta perbedaan penelitian ini dibandingkan penelitian sebelumnya dalam membangun *load balance server* yaitu sebagai berikut:

1. Nginx digunakan sebagai *load balancer* serta *backend server* yang menjalankan proses pada *http web server*.
2. Tidak menggunakan HAProxy sebagai *software* tambahan yang menjalankan *load balance* proses.

3. Ubuntu sebagai *operating system* dari *load balance server*, *backend server*, serta merupakan *operating system* dari penulis.
4. Proxmox sebagai *software* atau *tool* virtualisasi yang digunakan penulis dalam membuat *virtual image server* pada penelitian ini baik *load balance server* maupun *backend server*.

## 2.2 Landasan Teori

### 2.2.1 Blog Universitas Muhammadiyah Yogyakarta

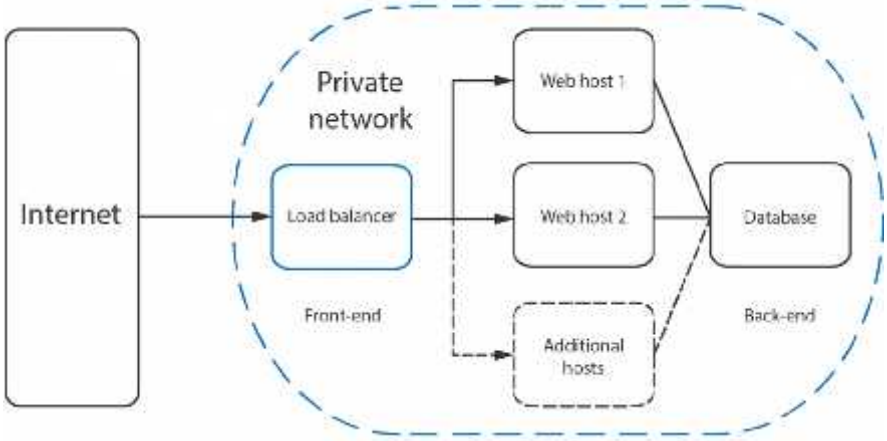
Universitas Muhammadiyah Yogyakarta (UMY) menggunakan web blog yang dibagun dengan memanfaatkan Wordpress sebagai CMS (*Content Management System*) terlihat pada gambar 2.2 dan 2.3. Wordpress dipilih dikarenakan *free* dan *open sources,* serta pengoperasiannya yang mudah. Blog UMY difungsikan sebagai sarana atau media diskusi antar pengguna blog, sebagai sarana pegumpulan tugas, artikel, maupun sebagai tempat memposting hasil dari penelitian, kegiatan akademis, kegiatan kampus, tutorial, dan sebagainya yang berhubungan dengan universitas. Sehingga blog menjadi media sarana yang dipergunakan untuk para jajaran aktivis akademisi maupun mahasiswa/i dalam mendukung kegiatan belajar mengajar pada universitas (http://blog.umy.ac.id).


**Gambar 2.2** Tampilan halaman depan blog UMY

**Gambar 2.3** *Login page* blog UMY

### 2.2.2 Load Balance

Nginx mendefinisikan *Load Balancing* atau *load balance* adalah proses pendistribusian beban terhadap sebuah *service* atau layanan atau aplikasi yang bertujuan untuk meningkatkan *resources, maximizing throughput, reducing latency, fail over*, serta memastikan aplikasi atau layanan tersebut mampu tersedia kapanpun *user* membutuhkan [11]. Cara kerja *load balance* pada umumnya terdapat dua jenis *server* yang masing-masing bertindak sebagai *load balance server* dan sebagai *back end server*. B*ack end server* berfungsi melayani *user interface* kepada *user*. Sedangkan *load balance server* yang akan mengarahkan kemana *request* atau aliran data *user* diarahkan dalam hal ini ke *back end server*. Struktur topologi dan mekanisme *load balance* dapat dilihat pada gambar 2.4 berikut.




**Gambar 2.4** Struktur topologi dan mekanisme *load balance*

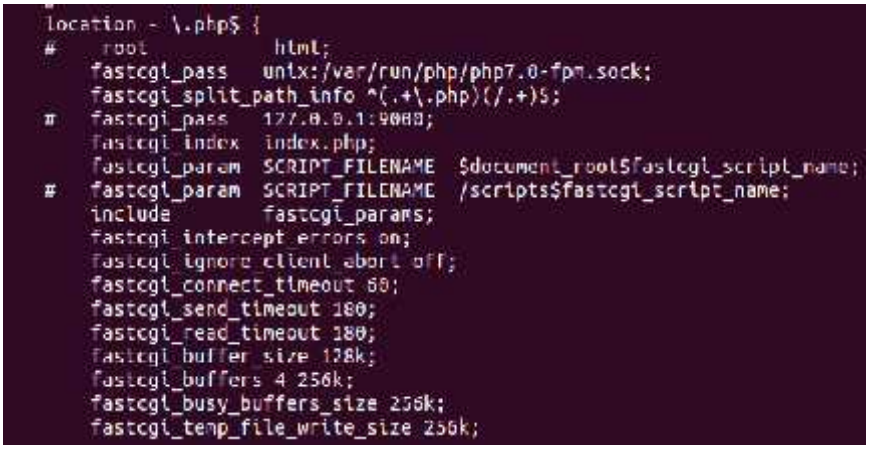
### 2.2.3 Nginx

Nginx (baca: Engine X), merupakan *high http* web *server*, yang bisa juga difungsikan sebagai *proxy*, *load balancer*, dan *HTTP cache*, yang dikembangkan pertama kali oleh Igor Sysoev yang berkewarganegaraan Rusia [12]. Nginx pertama kali diperkenalkan kepada publik pada tahun 2004, yang mana Nginx bersifat *free and open sources software,* yang artinya semua orang dapat dengan bebas mengembangkan serta mendistribusikannya, tetapi masih dengan menyertakan kode sumber atau berdasarkan dengan aturan dari *license* 2-*clause* BSD.

### 2.2.4 FastCGI Proxy

*FastCGI Proxy* merupakan *module* pada Nginx yang berfungsi mendukung mekanisme *load balance* dalam mendistribusikan *content* baik itu *content* dalam bentuk bahasa program ataupun dalam menterjemahkan permintaan *client* kepada *server*. Nginx menggunakan teknologi yang bernama *FastCGI Proxying* untuk menterjemahkan atau memprosess *request* atau permintaan dari *client* kepada *server*, sehingga permintaan *client* akan

dikerjakan oleh *FastCGI* [13]. *FastCGI*, atau *Fast Common Gateway Interface*, diakui oleh Nginx [13] merupakan cara yang terbaik dalam memaksimalkan kinerja baik dari sisi *server* maupun web *server* itu sendiri, dikarenakan *FastCGI* ini tidak melakukan pemisahan pada permintaan si *client*, yang nanti akan sangat berguna jika *content* yang akan didistribusikan bersifat dinamik. Secara lengkap konfigurasi pada *module FastCGI proxy* dapat terlihat seperti pada gambar 2.5 berikut.




**Gambar 2.5** Nginx php FastCGI

Terdapat fungsi-fungsi dari module *FastCGI proxy* yang mendukung proses *request/respond* antara *client* dan *server*, serta mendukung dalam penelitian *load balance* ini seperti yang ditunjukan pada gambar 2.5. Fungsi-fungsi tersebut akan dijelaskan pada subbab-subbab berikutnya.

**1. Fastcgi_pass**

*FastCGI_pass* merupakan *protocol* CGI yang berfungsi sebagai *socket* atau *proxy reverse* pada Nginx, yang digunakan dalam mekanisme *forwarding client request* kepada *backend server* [13].

**2. Fastcgi_param**

*Fastcgi_param, protocol* ini digunakan oleh Nginx ketika terdapat suatu *script* yang dieksekusi oleh *client* atau *user* dan butuh respon dari *server* itu sendiri, yang juga didefinisikan sebagai *variable* SCRIPT_FILENAME [13]. *Document_root* itu sendiri, mendefinisikan sebagai letak dari direktori utama didalam nginx, seperti pada gambar 2.5.

**3. Fastcgi_intercept_errors**

*fastcgi_intercept_errors* pada Nginx secara *default* memiliki fungsi sebagai tempat atau file yang secara berkala akan mencatat *error* jika terdapat kesalahan pada konfigurasi file *server block* maupun konfigurasi pada file *nginx.conf* [13]. File *error_log* tersebut *default*-nya terletak pada */var/log/nginx/*, dan Nginx sendiri secara default mematikan prosess *intercept* ini, yang artinya jika terjadi kesalahan tidak akan memunculkan ataupun merespon kepada *user* jika terjadi *error* pada halaman akan tetapi *user* akan diarahkan ke halaman utama dari sebuah website tersebut. Sedangkan *fastcgi_pass* merupakan *resolv proxy* yang akan menterjemahkan proses *request* dari *client* kepada *server*, akan tetapi jika ingin memodifikasi file serta letak *path* dari *fastcgi_intercept_error* tersebut bisa kita lakukan seperti pada gambar 2.6.

```
error_page 404 /errors/404.html;

location / {
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_intercept_errors on;
    ...
}

location /errors/ {
    # static
}
```

**Gambar 2.6** Nginx fastcgi_intercept_error [13]

**4. Fastcgi_ignore_client_abort**

Module *Fastcgi_ignore_client_abort* pada FastCGI *server* menunjukan, jika *client* menghentikan proses *request* terhadap *server*, maka *server* juga akan menghentikan respon kepada *client* tersebut, sehingga meminimalisir kemungkinan terjadinya DOS pada *server* [13]. Adapun kita bisa memodifikasi proses yang terjadi pada *server* dengan diantaranya apakah *server* akan melakukan redirect pada halaman lain, atau apakah akan melakukan *limit* terhadap koneksi tersebut.

**5. Fastcgi_connect_timeout**

*Fastcgi_connect_timeout* merupakan fungsi pada *server block* yang terdapat pada Nginx, yang berfungsi untuk memberikan batas respon per *server backend* dalam kasus *Load Balance* ini, jadi jika kita *setting fastcgi_connect_timeout* sebesar 60, maka lama respon yang dibutuhkan *server* untuk menanggapi atau merespon *request* per *client* tersebut adalah 60 detik, jika lebih dari itu maka biasanya akan muncul tulisan "**502 *Bad Gateway***" pada *browser client* [13].

**6. Fastcgi_buffer_size**

Ketika fitur *buffer* pada *fastcgi_buffer_size* digunakan, maka *server* akan menanggapi atau merespon lebih cepat dengan dengan menggunakan FastCGI *server buffer*, dikarenakan jumlah alokasi atau pemakaian *memory* pada *server* dapat diatur dengan sedemikian rupa sesuai dengan kebutuhan [13]. Untuk mengontrol hal tersebut biasanya *server administration* melakukan setting *fastcgi_buffer_size* yaitu sebesar 4K-8K, yang artinya setiap *server* akan mampu menampung *request* per *client* sebesar 4000 *megabits* sampai dengan 8000 *megabits* per-*second* tergantung dengan alokasi *memory* per-*server*. Ini juga bisa menjadi *alternative* seorang *sysadmin* jika ingin mengamankan *server*nya dari sisi Nginx terhadap *vulnerability buffer-offer flow.*

**7. fastcgi_send_timeout**

*Fastcgi_send_timeout* merupakan *module* pada Nginx FastCGI proxy, yang digunakan dalam menentukan batas dari *transmitting* sebuah *client request*, *default*nya adalah 60 detik [13], yang artinya jika paket yang berasal dari *client* gagal dalam *adjacent* (membentuk koneksi) pada server selama 60 detik atau lebih, maka paket akan di *drop* atau ditolak.

**8. Fastcgi_split_path_info**

*fastcgi_split_path_info* memiliki fungsi *Regular expression* atau lebih dikenal dengan *regex*, digunakan pada *module fastcgi_split_path_info* pada Nginx, yang berfungsi untuk menambahkan *http url* pada Nginx *web server* serta menunjukan *directory root access* pada Nginx [13]. Php *regex* juga berfungsi untuk medefinisikan

*path* atau letak dari *script* yang akan dieksekusi oleh Nginx berdasarkan *regular ekspresion rules*.

### 2.2.5 Least connected load balancing

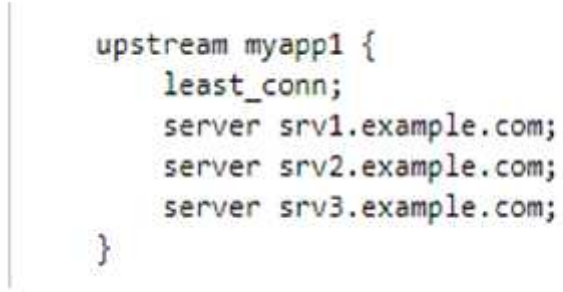
Terdapat empat metode *load balancing* yang dapa digunakan pada Nginx *web server*, yaitu [14]:

1. Round-robin (default Nginx load balancing)
2. Least connected
3. Session persistence (IP-Hash)
4. Weighted load balance

Secara *default* jika mengkonfigurasi tanpa ada masukan parameter tertentu, Nginx sudah menjalankan mekanisme *load balance* yaitu dengan menggunakan metode *round-robin*. Metode kedua sekaligus metode *load balance* yang digunakan pada penelitian ini yaitu *least connected load balance*. *Least connected load balance* merupakan salah satu metode *load balance* dimana beban *server* akan didistribusikan secara acak mengacu pada *active session time per client* [15].

Terdapat beberapa perbedaan mendasar antara *round-robin* dan *least connected load balance* diantaranya mengenai pembagian *active session* pada *client*, jika pada *least connected server load balance* akan melakukan secara acak terkait dengan pembagian *session per client address*, artinya, antara *client* satu dan lainnya akan mendapatkan *session* yang tidak sama (acak). Sedangkan pada *round-robin*, tidak terdapat metode *session per client address* tersebut [15].

Fitur dari *load balance* yang digunakan didalam penelitian ini dapat terlihat seperti pada contoh gambar 2.7, *module name* (*upstream*) didefinisikan dengan nama "myapp1". *Least connection* artinya *server* yang menjalankan *module* ini akan mendistribusikan *http request client* secara merata dan acak terhadap *backend server* yang didefinisikan dengan parameter *server* [14]. Contoh dari penerapannya terdapat pada gambar 2.7 berikut.




**Gambar 2.7** Least connected load balance [15]

Terdapat *module upstream* yang digunakan dalam mendistribusikan *x-http request client* kepada *backend server* yang bernama *myapp1* seperti yang ditunjukan pada gambar 2.7.

### 2.2.6 Virtualization

Teknologi yang digunakan sekarang sudah masuk kedalam era *virtualization*. Virtualization atau virtualisasi adalah sebuah proses membuat sebuah virtual *server* atau *software* dengan menggunakan *software* utilitas tertentu seperti proxmox, vmware, dan lainnya, sehingga didalam satu fisik *server* terdapat lebih dari satu virtual *server* atau virtual aplikasi [16]. Proxmox digunakan sebagai alat virtualisasi *free* dan *open source*

pada penelitian ini, yang berfungsi untuk membuat *virtual image server load balance* maupun *backend server*.

### 2.2.7 Proxmox

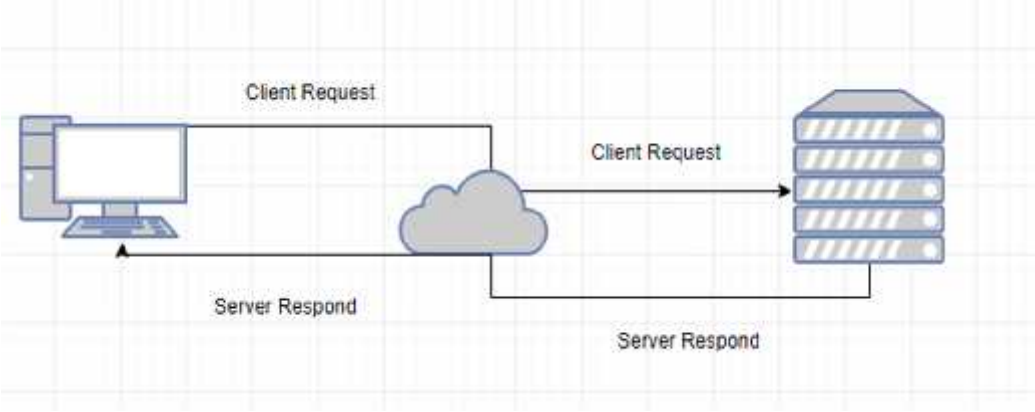
Proxmox adalah *baremetal hypervisor* atau yang biasa dikenal dengan sebuah *platform* virtualisasi yang dipasang atau diinstall langsung diatas *server* [17]. Proxmox memiliki lisensi *open sources* yang artinya siapa saja dapat menggunakan, mendistribusikan, dan menyebarluaskan proxmox itu sendiri. Proxmox merupakan hasil dari pengembangan dari Debian *project* yang telah dimodifikasi dan dipadukan dengan *kernel* 4.4 LTS dari Ubuntu. Beberapa keuntungan menggunakan proxmox diantaranya [18]:

1. *Open Sources,* yang artinya tanpa dikenakan biaya jika ingin memakainya.
2. Menggunakan KVM dan LXC technologies, KVM sendiri merupakan fitur dimana kita dapat menjalankan banyak *private images* dengan *private* virtualisasi *hardware* didalam satu *server*, dimana *private server* satu dan *private server* lainnya memiliki *network segment* dan kapasitas penyimpanan atau *hard disk* tersendiri. Sedangkan LXC merupakan fitur dimana kita bisa membuat banyak linux *container* dengan satu fisik *server*, dan tiap-tiap *container* itu memiliki konfigurasi baik dari network maupun keamanan yang berbeda. Proxmox juga lebih memanjakan *user* yang lebih memahami *GUI interfaces* dari pada *CLI interfaces*, oleh kerena itu proxmox di *design* menggunakan antar muka GUI.
3. Proxmox juga menyediakan fitur berupa *backup* otomatis dimana kita bisa melakukan *backup* sesuai dengan keinginan kita dan hasil *backup* juga bisa dikirim melalui e-mail pribadi kita sebagai *system administrator*.

### 2.2.8 Server

Didalam dunia komputer, *server* memiliki arti yakni, sebuah computer program atau bisa disebut juga induk atau pusat dari komputer-komputer yang ada, yang memiliki fungsi sebagai penyedia atau wadah untuk berbagai macam aplikasi atau layanan yang nantinya dipergunakan untuk melayani *user* [19].

Didalam *server* itu sendiri, memiliki banyak fungsi yang disebut *service*, seperti *server* yang memiliki *service* sebagai *mail, proxy, web server*, dan lain-lain. *Service* pada *server* sendiri mampu melayani banyak *user* sekaligus dalam waktu yang bersamaan. *Server* sendiri dalam berkomunikasi dengan *client* menggunakan protokol *request-response*, seperti yang digambarkan pada gambar 2.8.




**Gambar 2.8** Client- *Server* Communication

*Server computer* telah banyak mengalami perubahan dari masa ke masa, dari perubahan bentuk dan ukuran sampai perubahan mobilitas dalam artian *server* sekarang hampir semuanya sudah berbasis virtual bahkan sampai dengan yang terbaru sekarang yakni berbasis *cloud*, yang semakin memudahkan seseorang memiliki *server* diberbagai

negara tanpa perlu memikirkan resources yang dibutukan oleh *server* baik fisik *server* maupun ruang atau *space* untuk menampung fisik dari *server*.

Gambar 2.9 berikut ini merupakan contoh dari *server* yang ukurannya bisa dikatakan kecil atau *micro*.

**Gambar 2.9** IBM *microserver* [20]

Beberapa *vendor software* dan teknologi seperti IBM, HP, Dell, Sun, Rainer, Extron dan TrendMicro, dan lain-lain, juga terus mengembangkan jenis-jenis *product server* mereka masing-masing yang siap bersaing secara global.

### 2.2.9 *Server* Hardening

Berasal dari kata *Hardening*, yang jika didalam dunia *digital* atau komputer yaitu, sebuah proses dimana seorang *system administrator* mengamankan *system* yang dia pegang atau jalankan, mengamankan yang dimaksud adalah, meminimalisir seminimal mungkin kemungkinan-kemungkinan celah atau *vulnerable* yang bisa dimanfaatkan *attacker* dalam hal ini *cracker* untuk masuk kedalam *system* yang telah dibangun [21]. Terdapat beberapa cara yang digunakan dalam melakukan hardening pada server, antara lain:

1. Mematikan service token pada Nginx
2. Menambahkan fungsi http anti-XSS module OWASP pada Nginx
3. Melakukan limit pada koneksi per IP Address

### 2.2.10 Ubuntu

Ubuntu merupakan salah satu *operating system* distribusi Linux yang berbasiskan Debian dan didistribusikan sebagai perangkat lunak bebas [22]. Ubuntu dirancang untuk kepentingan penggunaan pribadi, namun versi server Ubuntu juga tersedia, dan telah dipakai secara luas.

Proyek Ubuntu resmi disponsori oleh *Canonical Ltd*, yang merupakan sebuah perusahaan yang dimiliki oleh pengusaha Afrika Selatan Mark Shuttleworth [22]. Tujuan dari distribusi Linux Ubuntu adalah membawa semangat yang terkandung di dalam filosofi Ubuntu ke dalam dunia perangkat lunak. Ubuntu adalah sistem operasi lengkap berbasis Linux, tersedia secara bebas, dan mempunyai dukungan baik yang berasal dari komunitas maupun tenaga ahli professional.

Pada penelitian kali ini, menggunakan dua versi Ubuntu, yang pertama yaitu versi 16.04 LTS (Long Term Support/Service) yang digunakan sebagai *operating system* pada laptop penulis, dan versi Ubuntu 14.04 LTS yang merupakan *operating system* pada *backend server* dan *load balance server*.

### 2.2.11 Nano

Nano merupakan *editor text* berbasis *command line* (CLI) pada terminal GNU/Linux [23]. Nano berfungsi sebagai *text editor* yang dipergunakan untuk melakukan *editing code* pada penelitian ini.

### 2.2.12 PHP 7.0

Php merupakan salah satu bahasa program yang berguna sebagai *server scripting* [24]. Php 7.0 berfungsi sebagai *server side scripting* yang digunakan oleh Nginx dalam menterjemahkan *client* proses pada Wordpress. Blog yang digunakan dalam penelitian kali ini menggunakan Wordpress yang menggunakan php 7.0 sebagai bahasa pemrograman utama.

### 2.2.13 Php-fpm

Nginx menggunakan php-fpm untuk menjalankan beberapa fungsi diantaranya yaitu, sebagai penterjemah dari php sebagai *server side scripting* khususnya dalam bahasa pemrograman php, sebagai pengatur *user access permission*, serta bisa juga difungsikan sebagai php *hardening* [25].

Tidak seperti kompetititor atau saingan dari Nginx yaitu Apache, jika pada Apache dalam menangani atau meng-*handle* prosess yang membutuhkan bahasa pemrograman php, Apache menggunakan *mod_php module* bawaan dari Apache itu sendiri, jika pada Nginx dalam menangani proses yang membutuhkan bahasa pemrograman khususnya php, Nginx menggunakan *php-fpm* yang diinstall secara terpisah dari Nginx itu sendiri, php-fpm ini terdapat pada module php. Secara teknisnya Nginx dalam melakukan proses *forwarding client request* ke *server*, itu menggunakan *proxy* yang terdapat pada Nginx itu sendiri. Jika pada Nginx sendiri dinamakan *fastcgi_pass*.

### 2.2.14 Wordpress

Wordpress merupakan *free and open-source software content management system* (CMS), yang dibangun berdasarkan bahasa pemrograman php dan mysql [26]. Wordpress merupakan CMS *default* yang digunakan oleh blog UMY.

### 2.2.15 Putty Ssh Client

Putty merupakan alat untuk menghubungkan *client* dan *server*, atau disebut juga dengan ssh *client*, putty difungsikan sebagai alat atau tool yang digunakan sebagai penghubung penulis dengan *server-server* yang akan dikonfigurasi dalam penelitian ini [27].

### 2.2.16 MariaDB Database

MariaDB merupakan salah satu *database open sources* yang dikembangkan dari MySQL database [28]. Blog UMY menggunakan MariaDB sebagai *database server*, yang dipergunakan untuk menyimpan *data user*, artikel, gambar, dan lainnya. Pada penelitian kali ini juga menggunakan MariaDB sebagai *database server*.

Sebelum berganti nama menjadi MariaDB, dahulunya MariaDB merupakan MySQL, hal ini terlihat dari syntax-sytax pada MariaDB yang hampir semuanya mirip dengan syntax pada MySQL. Sebelum mengganti nama menjadi MariaDB, MySQL dahulunya masih dibawah naungan atau diurus berdasarkan komunitas, setelah Oracle membeli MySQL, barulah berganti nama menjadi MariaDB.