JURNAL REKAYASA SISTEM DAN INDUSTRI
e-ISSN: 2579-9142 p-ISSN: 2356-0843
http://jrsi.sie.telkomuniversity.ac.id

Implementation Autoscaling Container Web Server using Kubernetes Promox-Based on Server University of Darussalam Gontor

Imron Rosyadi*, Shoffin Nahwa Utama, Oddy Virgantara Putra

Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Darussalam Gontor

INFO ARTIKEL

Article history:
Diterima 03-03-2019
Diperbaiki 21-04-2019
Disetujui 15-05-2019

Kata Kunci:
Virtualisasi, *Autoscaling*, Proxmox, Kubernetes, Kontainer

ABSTRAK

Layanan web server berbasis virtualisasi di Universitas Darussalam Gontor belum diimplementasikan, selain itu permasalahan website pada umumnya yaitu web server tidak mampu melayani jumlah permintaan yang tinggi sehingga web server bisa down. Salah satu permasalahan tersebut disebabkan oleh resource server yang kurang mumpuni dan pengelolahan web server yang kurang optimal. Masalah tersebut dapat diatasi secara efektif dan efisien dengan virtualisasi server dan autoscaling. Dimana virtualisasi server dikelola Proxmox dan autoscaling dikelola Kubernetes. Proxmox merupakan platform virtualisasi yang bersifat opensource dan didukung oleh hypervisor KVM dan OpenVZ serta dikelola dengan mudah. Sedangkan Kubernetes termasuk platform yang berfungsi sebagai container orchestration untuk melakukan penjawalan, monitoring, scaling, dan recovery kontainer. Kolaborasi virtualisasi dan autoscaling dapat melayani akses website secara efektif dan efisien. Metodologi penelitian yang digunakan adalah clustering dengan menerapkan konsep high availability dimana terdapat secondary-server sebagai backup-server untuk primary-server. Hasil dari penelitian ini mencakup tiga parameter yaitu Throughput, Response Time, dan CPU Usage. Hasil akhir menunjukkan bahwa web server yang menerapkan virtualiasi dan autoscaling lebih baik daripada web server yang tidak menerapkan virtualisasi dan autoscaling dengan selisih nilai parameter Throughput sebesar 4494.34 KB/s, selisih nilai parameter Response Time sebesar 14,46 requests/detik, dan selisih nilai parameter CPU Usage sebesar 13.01%.

Keywords:
Virtualization, Autoscaling, Proxmox, Kubernetes, Container

ABSTRACT

Web server services based on virtualization at the University of Darussalam Gontor have not yet been implemented, besides the problem of websites in general is that the web server is not capable of serving a high number of requests so the web server can be down. One of these problems is caused by a server resource that is not capable and the management of a web server that is less than optimal. These problems can be addressed effectively and efficiently with server virtualization and autoscaling. Where server virtualization is managed by Proxmox and autoscaling managed by Kubernetes. Proxmox is a virtualization platform that is open source and is supported by KVM and OpenVZ hypervisors and is easily managed. Whereas Kubernetes is a platform that functions as a container orchestration to carry out the appointment, monitoring, scaling and recovery of containers. Collaboration virtualization and autoscaling can serve website access effectively and efficiently. The research methodology used is clustering by applying a high availability concept where there are secondary-servers as backup-servers for primary-servers. The results of this study include three parameters, namely Throughput, Response Time, and CPU Usage. The final results show that web servers that implement virtualization and autoscaling are better than web servers that do not implement virtualization and autoscaling with the difference in Throughput parameter values of 4494.34 KB / s, the difference in Response Time parameter values of 14.46 requests / second, and the difference in parameter values CPU Usage is 13.01%.

1. Pendahuluan

Perkembangan teknologi informasi dan komunikasi semakin lama semakin canggih, hal itu dapat diketahui dari kemudahan akses informasi melalui *website*. *Website* merupakan sebuah kumpulan halaman-halaman web beserta berkas-berkas pendukungnya, seperti berkas gambar, video, dan berkas digital lainnya yang disimpan pada sebuah web

DOI: https://doi.org/10.25124/jrsi.v6i02.362

*Penulis korespondensi
Email: imron.rosyadi@unida.gontor.ac.id (Rosyadi, I.), shoffin@unida.gontor.ac.id (Utama, S.N.), oddy@unida.gontor.ac.id (Putra, O.V.)

server yang umumnya dapat diakses melalui internet. Pengertian lain menyebutkan bahwa, *website* merupakan sekumpulan *folder* dan berkas yang saling berintegrasi yang mengandung banyak perintah dan fungsi tertentu, seperti fungsi daripada tampilan aplikasi, fungsi untuk menyimpan data, dan lain sebagainya [1].

Kinerja website akan bagus apabila penyediaan infrastruktur server yang baik, seperti spesifikasi server dan pengelolahannya. Pengelolahan yang baik bisa berupa virtualisasi server. Virtualisasi merupakan teknologi yang memungkinkan sebuah server fisik dapat dipakai secara bersama – sama dalam beberapa layanan [2]. Layanan server yang dimaksud seperti web server, database server, *firewall*, dan lain sebagainya [13]. Selain itu konfigurasi web server juga perlu dilakukan, semakin baik konfigurasi web server maka layanan *website* semakin baik pula.

Berdasarkan informasi dari staf Pusat Pelayanan Teknologi Informasi dan Komunikasi (PPTIK) Universitas Darussalam Gontor (UNIDA Gontor) masih belum memanfaatkan adanya layanan server berbasis virtualisasi seperti web server. Layanan website di lingkungan Perguruan Tinggi pada khususnya sangat dibutuhkan karena *website* merupakan pusat sistem informasi yang berisi berita seputar kegiatan akademik maupun non – akademik. Selain itu permasalahan akses layanan *website* pada umumnya adalah kemampuan web server dalam menangani permintaan dari pengguna seperti menerima *request* dan mengirim *response* HTTP. Permasalahan tersebut dapat diatasi dengan *autoscaling*. Menurut A. Bondi *autoscaling* adalah penambahan perangkat pada sistem server tanpa mengganggu kinerja server lainnya. Hal ini dilakukan untuk mengatasi beban kerja yang belum bisa dilakukan server [3].

Virtualisasi server menggunakan platform *Proxmox Virtual Environment (PVE)* karena *PVE* bersifat *opensource* dan memiliki performa yang stabil dengan kelas *enterprise*. *PVE* memiliki *hypervisor Kernel based Virtual Machine (KVM)* dan *OpenVZ*. *KVM* merupakan virtualisasi server berbasis mesin virtual dan *OpenVZ* merupakan virtualisasi server berbasis kontainer. Selain itu *PVE* bisa dioperasikan dengan mudah berbasis web [4]. Sedangkan *autoscaling* menggunakan Kubernetes. Kubernetes merupakan salah satu platform yang berfungsi untuk orkestrasi kontainer, termasuk didalamnya terdapat fitur *Container Deployment, Persisten Storage, Container Health Monitoring, Compute Resource Management, Autoscaling, dan High Availability* [5].

Metode yang digunakan dalam penelitian ini yaitu *clustering* dengan menerapkan konsep h*igh availability*, dimana terdapat *secondary-server* sebagai *backup-server* dari *primary-server*. Apabila terdapat kendala pada *primary-server*, maka layanan tetap berjalan karena layanan server ditampung oleh *secondary-server.* Berdasarkan penelitian sebelumnya yaitu penelitian pertama yang dilakukan oleh Didik Sudyana, Edwar Ali dengan judul Virtualisasi Server dengan Proxmox untuk Pengoptimalisasian Penggunaan *Resource* Server pada UPT Teknologi dan Komunikasi Pendidikan, penelitian tersebut memanfaatkan virtualsiasi server saja menggunakan Proxmox dengan fitur *Live Migration* [6]. Kemudian penelitian kedua dilakukan oleh Michel Tighe dan Michael Bauer dengan judul *Integrating Cloud Application Autoscaling with Dynamic VM Allocation* yang menerapkan *autoscaling* dengan algoritma *First Heuristic Fit*, dimana algoritma tersebut dapat mengalokasikan mesin virtual secara dinamis terhadap server yang membutuhkan *resource* tambahan [7]. Selanjutnya penelitian ketiga dilakukan oleh I Gede Putu Krisna Juliharta, Wayan Supedena, dan Dandy Paramana Hostiadi dengan judul *High Availability* Web Server berbasis *Opensource* dengan Teknik *Failover Clustering* yang memanfaatkan *high availability* untuk kebutuhan layanan server dengan *tool Distributed Replicated Block Device (DRDB)* yang berkoordinasi dengan *heartbeat* untuk menentukan jalur server yang aktif [8]. Selanjutnya penelitian keempat dilakukan oleh Yosua Tito Sumbogo, Mahendra Data, dan Reza Andria Siregar dengan judul Implementasi *Failover* dan *Autoscalling* Kontainer Web Server Nginx pada Docker menggunakan Kubernetes yang memanfaatkan kubernetes untuk orkestrasi kontainer web server, dan melakukan uji web server untuk membandingkan performa web server menggunakan *autoscaling* dan web server yang tidak menggunakan *autoscaling*, akan tetapi penelitian ini hanya menguji *homepage* dari web server saja[9]. Penelitian kelima dilakukan oleh Wito Delnat, Eddy Truyen, Ansar Rafique, Dimitri Van Landuyt, dan Wouter Joosen dengan judul *K8-Scalar: a Workbench to Compare Autoscalers for Container-Orchestrated Database Clusters*, penelitian tersebut melakukan *autoscaling* dengan kubernetes yang diterapkan pada *database-server* Cassandra [10]. Berdasarkan referensi penelitian terdahulu maka penelitian ini mengkolaborasikan layanan virtualisasi server dengan *autoscaling* dan melakukan pengujian performa web server dengan menerapkan layanan *website* berbasis *Content Management System (CMS)* yaitu wordpress dengan *database* mysql.

Hasil yang diharapkan dari penelitian ini adalah adanya layanan web server yang dapat melayani akses *website* dengan baik dan mengurangi beban server akibat lonjakan trafik permintaan *client* yang tinggi selain itu menghemat biaya infrastruktur server dan mengurangi energi listrik.

2. Metode Penelitian

2.1 Analisis Kebutuhan

1. Perangkat Keras *(Hardware)*

a. *Server* Dell R230

i. Xeon E3-1225 v6 3.36 *Ghz*

ii. 8GB UDIMM

iii. 1TB 7.2K RPM SATA 3.5”

b. Kabel *Unshielded Twisted Pair (UTP)*

c. Konektor RJ45

d. Switch

e. Router Mikrotik

2. Perangkat Lunak *(Softwa**re)***

a. Proxmox VE

b. Ubuntu Server 18.04

c. Wordpress 4.8

d. Mysql 5.6

e. Gliffy

f. NGINX

g. Kubernetes

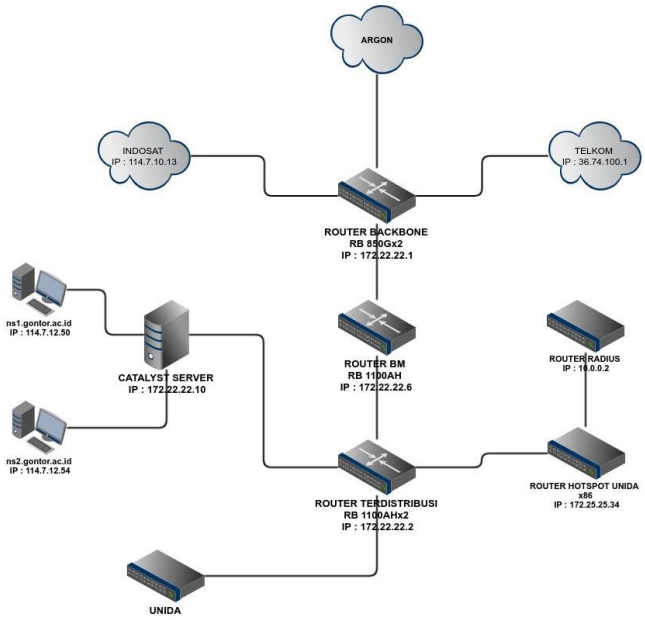
h. Apache Benchmark

i. Httperf

j. *Sar*

2.2 Topologi Jaringan UNIDA Gontor

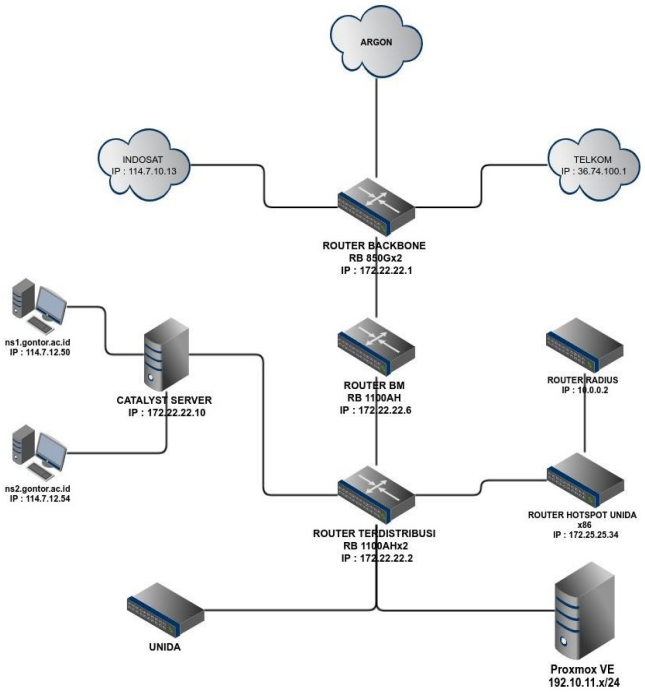
Topologi jaringan adalah suatu cara yang dapat menghubungkan komputer dengan perangkat jaringan. Cara yang digunakan berbeda-beda sesuai dengan aturan topologi jaringan yang sudah diterapkan. Macam-macam aturan topologi jaringan tersebut diantaranya topologi *bus, ring, star, tree, dan mesh*. Masing- masing topologi tersebut memiliki konsep yang berbeda-beda, memiliki kelebihan dan kekurangan masing-masing. Maka dari itu diperlukan kecermatan untuk menerapkan topologi tersebut kedalam studi kasus yang ada [11]. Adapun topologi jaringan yang digunakan UNIDA Gontor saat ini menggunakan tree dan layaan server masih menerapkan layanan *Domain Name Server (DNS)* untuk melayani pembuatan domain *website*. Topologi jaringan yang digunakan UNIDA Gontor saat ini dapat dilihat pada Gambar 1.



Gambar 1. Topologi Jaringan UNIDA Gontor Sekarang

2.3 Topologi Jaringan Usulan

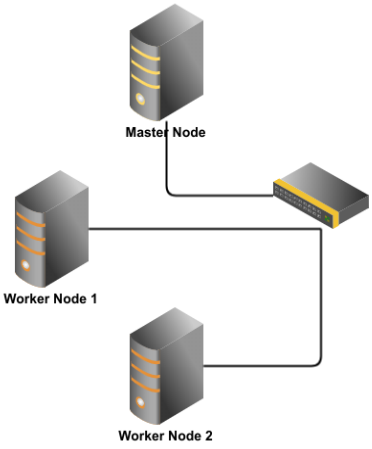
Penelitian ini menambahkan layanan web server berbasis virtualisasi agar *website* bisa dikelola dengan baik. Adapun virtualisasi tersebut menggunakan platform *PVE* yang berfungsi sebagai *orchestrator* dalam menangani mesin virtual. Fungsi dari mesin virtual sebagai *guest operating system* yakni menampung aplikasi beserta paket dependensinya sehingga bisa berjalan dengan baik. Tidak hanya virtualisasi yang diterapkan untuk mengelola *website* tersebut akan tetapi akan lebih efisien apabila dikolaborasikan dengan *autoscaling* untuk mengelola *skalabilitas* web server yang ditunjukkan pada Gambar 2.



Gambar 2. Topologi Jaringan UNIDA Gontor Usulan

2.4 Topologi Jaringan Cluster-Server

Metode yang yang digunakan dalam penelitian ini yaitu *clustering* yang menerapkan konsep *High Availability*. *High Aavailability* merupakan teknik replikasi terhadap server utama apabila server utama terjadi kendala. Server replikasi *(backup-server)* memiliki layanan yang sama seperti server utama, hal ini bertujuan untuk mengatasi apabila server utama mengalami kendala dengan memindahkan replika sistem ke *backup-server* tanpa mengalami *down time.* Konsep *high availability* tersebut dapat disajikan dalam bentuk Gambar 3.



Gambar 3. Konsep *High Availability*

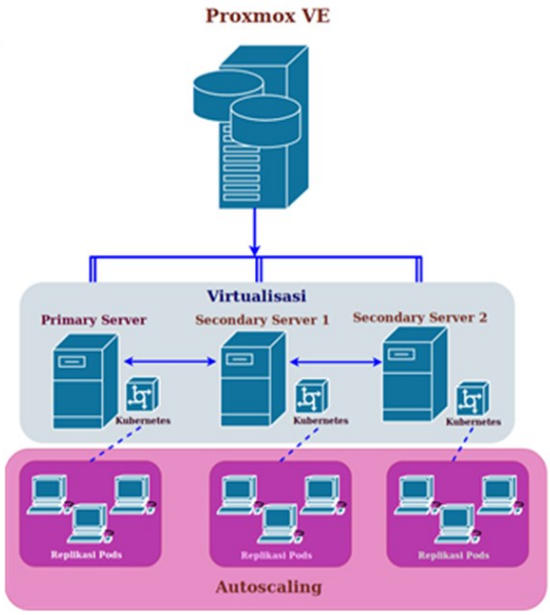
Adapun masing-masing mesin virtual memiliki spesifikasi yang berbeda baik pada *primary-server* maupun *secondary-server*. Sistem operasi yang digunakan yaitu Ubuntu server 18.04. Spesifikasi pada masing-masing mesin virtual yang ditunjukkan pada Tabel 1.

Tabel 1.
Spesifikasi Mesin Virtual

No.	Mesin Virtual	Spesifikasi
1	*Primary-Server*	• Prosesor : 2 Core • RAM : 4 GB • Hardisk : 32 GB
2	*Secondary-Server*	• Prosesor : 1 Core • RAM : 2 GB • Hardisk : 32 GB

2.5 Toplogi Jaringan Virtualisasi dan Autoscaling

Kolaborasi virtualisasi dengan *autoscaling* dapat meningkatkan *resource* server dan *skalabilitas* pengelolahan *website*. Konsep *autoscaling* memiliki kemampuan untuk menyesuaikan sumber daya yang dimiliki server seperti menurunkan atau menambah jumlah proses sesuai dengan kebutuhan sistem tanpa mengganggu proses yang sedang berjalan. Sedangkan konsep dari virtualisasi berfungsi untuk meningkatkan jumlah *resource* yang tidak terpakai untuk menambah kinerja dari proses yang berjalan, sehingga tidak ada *resource* server yang terbuang percuma. Kedua konsep apabila dikolaborasikan dapat mengelola layanan server khususnya web server menjadi lebih efektif dan efisien. Apabila terjadi lonjakan permintaan *website* yang tinggi dan *resource* yang dimiliki server tidak mampu menangani permintaan tersebut, maka *resource* mesin virtual dapat ditingkatkan oleh Proxmox *VE*. Konsep tesebut dapat dijelaskan pada Gambar 4.

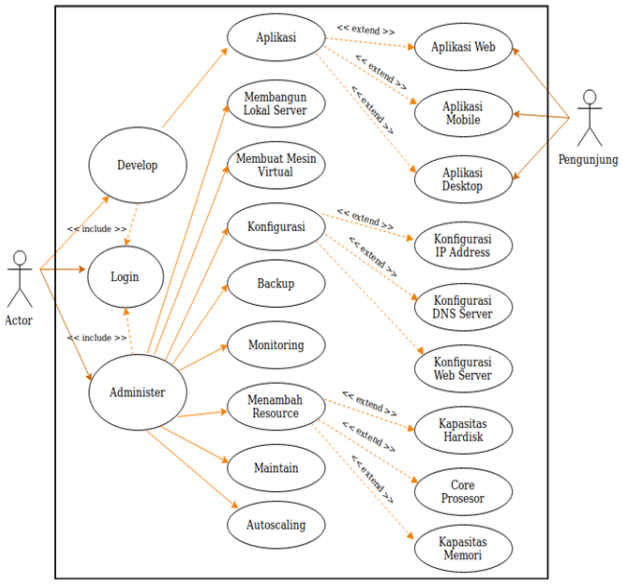


Gambar 4. Desain Topologi Jaringan Virtualisasi dan Autoscaling

2.6 Sistem Permodelan

2.6.1 Use Case Diagram

Use case diagram yang digunakan dalam penelitian ini dapat disajikan pada Gambar 5.



Gambar 5. Use Case Diagram Kolaborasi Virtualisasi dan Autoscaling

2.6.2 Activity Diagram

Activity diagram yang digunakan dalam penelitian ini dapat disajikan pada Gambar 6.

3. Hasil dan Pembahasan

3.1 Implementasi Sistem

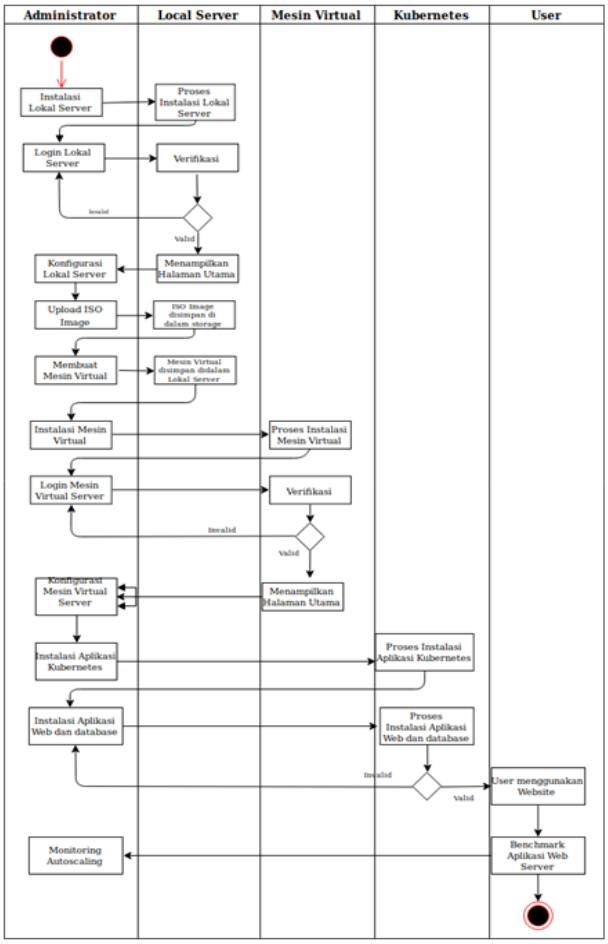
Penelitian ini menggunakan platform Proxmox VE untuk mengelola mesin virtual. Adapun mesin virtual yang dikelola berjumlah tiga mesin virtual, dimana mesin virtual satu berfungsi sebagai *master node* dan kedua mesin virtual lainnya berfungsi sebagai *worker node*. Tampilan Proxmox VE yang digunakan untuk mengelola ketiga mesin virtual tersebut dapat dilihat pada Gambar 7.

Pada Gambar 7 menunjukkan bahwa *Identificaton Number (ID)* mesin virtual 106 sebagai *master node* dan *ID* 101 dan 103 sebagai *worker node*. Pada setiap mesin virtual diinstal aplikasi kubernetes seperti *kubectl, kubeadm, kubelet* dan aplikasi docker. Docker digunakan untuk membaca aplikasi kontainer sedangkan kubernetes untuk orkestrasi kontainer. Selanjutnya dilakukan konfigurasi *hostname* dan *ip address* pada masing-masing mesin virtual untuk membuat jaringan *cluster*. Konsep kubernetes menerapkan jaringan *cluster* untuk *backup-server,* apabila *master node* mengalami gangguan layanan server bisa dikelola oleh *worker node*.

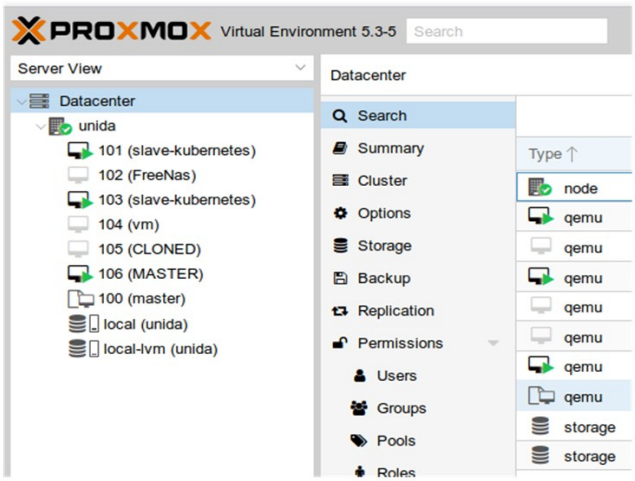
Adapun *website* yang digunakan untuk pengujian *autoscaling* menggunakan *website* berbasis *CMS* wordpress dengan *database* mysql. Berbeda dengan *autoscaling* pada web server layanan Nginx biasa, karena akses layanan *website* lebih berat. Adapun website berbasis *CMS* wordpress dapat dilihat pada Gambar 8.

Setelah pembuatan *website* berhasil dilakukan, langkah selanjutnya adalah pengujian *autoscaling* pada web server. Konsep *autoscaling* menggunakan *horizontal pod autoscaling (hpa)* dimana menambah layanan *pod* web server secara otomatis apabila terdapat permintaan yang tinggi sehingga melebihi target CPU *pod* yang ditentukan. Dalam penelitian ini menggunakan target CPU *pod* sebesar 50%. Target CPU

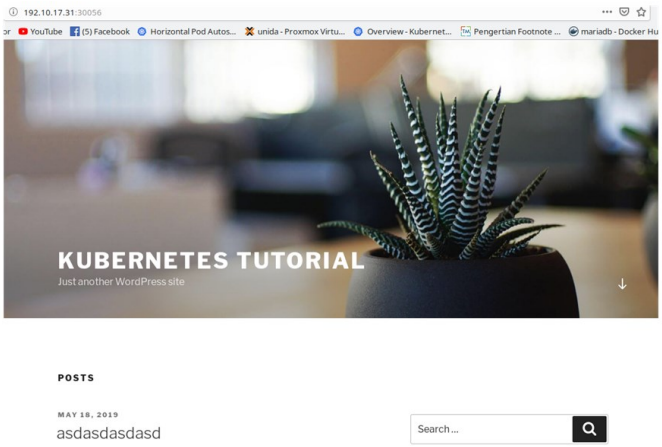
tersebut ditentukan berdasarkan karakter aplikasi, apabila aplikasi memiliki waktu *warming up* atau *startup* yang lama maka target CPU *pod*-nya lebih tinggi begitupun sebaliknya. Hal ini digunakan untuk menghindari terjadinya kelambatan aplikasi untuk mulai bekerja.



Gambar 6. Activity Diagram Virtualisasi dan Autoscaling



Gambar 7. Tampilan Awal Proxmox VE



Gambar 8. Website berbasis *CMS* Wordpress

Waktu yang ditentukan oleh kubernetes pada saat *scaling up pods* membutuhkan waktu 30 detik secara *default* untuk mempersiapkan *pod* sampai bisa digunakan, namun apabila *pod* tersebut mengalami kendala karena jumlah *resource* yang kurang maka waktu yang dibutuhkan untuk men-*deploy pod* yaitu 5 menit secara *default*. *Horizontal pod autoscaling* yang terdapat pada kubernetes memiliki algoritma dalam mengatur jumlah *instance* atau *pod* yang di-*deploy*. Men-*deploy instance* dapat diatur sesuai dengan jumlah minimal dan maksimal *instance*. Hal ini berkaitan dengan spesifikasi *hardware* CPU pada *server* sehingga *instance* yang di-*deploy* tidak melebihi batas spesifikasi CPU *server*. Algoritma tersebut dijelaskan pada rumus berikut. [12].

Algoritma Horizontal Pod *Autoscaling* (1)

Input: U_{target}, ***ActivePods***

// Target *utilization* and the set of *active Pods*

Output: P // The *target number* of *Pods* to *deploy*

1. **while true do**
2. **for all** $i \in ActivePods$ do
3. $U_i = getRelativeCPUUtilization(i);$
4. $U = U \cup \{U_i\}$
5. **end for**
6. *P = ceil(sum(U) / U_{target});*
7. *wait π // wait π seconds, the control loop period*
8. **end while**

Algoritma tersebut menunjukkan nilai input berisi U_{target} (target *Utilization*) atau persentase pemakaian CPU dan *Active Pods* yaitu jumlah *pods* yang aktif. Sedangkan *output* disimbolkan dengan huruf P yang berarti jumlah target *pod* yang di-*deploy*. Selama melakukan *scaling pod* disimpan terlebih dahulu pada baris bernomor 3 dan 4. Setelah itu jumlah *pod* akan ditentukan pada baris 5 sesuai dengan perhitungan rumus berikut.

$$P = \left\lceil \frac{\sum i \in ActivePods U_i}{U_{target}} \right\rceil \quad (2)$$

Algoritma *autoscaling* pada kubernetes dapat menentukan jumlah *pod* yang di-*deploy* pada saat *scaling up*, misalnya terdapat tiga *pods* yang berjalan setelah *scaling up* dengan konsumsi CPU yang berbeda-beda yaitu 79%, 75%, dan 83%. Sedangkan target *utilization* sebesar 50%. Jadi jumlah *instance pods* yang di-*deploy* berjumlah 5 *pod*. Nilai 50% itu merupakan target CPU diberikan kepada setiap *pod*. Apabila terjadi permintaan yang melebihi batas target *pod*, maka kubernetes akan melakukan replikasi melalui objek *replica set* sesuai dengan jumlah permintaan *client*.

3.1.1 Uji Coba

Adapun pengujian autoscaling terhadap web server pada penelitian ini menggunakan tiga parameter yaitu throughput, response time, dan CPU usage. Dimana ketiga parameter tersebut saling berkaitan dan sesuai apabila dijadikan sebagai parameter untuk menguji performa web server. Pada penelitian ini menggunakan skenario pengujian dengan jumlah koneksi dan tingkat konkurensi yang disajikan pada Tabel 2.

Tabel 2.
Jumlah Koneksi dan Konkurensi

No.	Total Koneksi	Konkurensi (Request/detik)
1	10000	100
2	20000	200
3	30000	300
4	40000	400
5	50000	500

Berikut skenario pengujian yang dilakukan untuk *benchmark* web server:

a. Pengujian performa web server *atau benchmark* web server dilakukan terhadap server yang menerapkan virtualisasi dan *autoscaling* (sesudah konfigurasi) dengan server yang tidak menerapkan keduanya (sebelum konfigurasi).
b. Pengujian pertama dilakukan untuk mendapatkan kriteria *throughput* dengan cara menentuka jumlah total koneksi sebesar 10000 koneksi terlebih dahulu dengan jumlah konkurensi mulai dari 100 *requests*/detik, apabila jumlah konkurensi sudah mencapai 500 *requests*/detik maka diganti dengan jumlah koneksi 20000 koneksi dengan tingkat konkurensi secara berkala seperti sebelumnya, pengujian dilakukan sampai pada jumlah koneksi mencapai 50000 koneksi.
c. Pengujian kedua dilakukan untuk mendapatkan kriteria *response time* dengan menentukan jumlah koneksi dan jumlah konkurensi seperti sebelumnya.
d. Pengujian ketiga dilakukan untuk mendapatkan kriteria CPU *usage* dengan menentukan jumlah koneksi dan jumlah konkurensi seperti sebelumnya.

Metode analisis yang digunakan untuk membahas hasil pengujian tersebut menggunakan metode deskriptif. Metode deskriptif adalah metode yang digunakan untuk membandingan hasil pengujian web server sebelum dan setelah konfigurasi dengan parameter *throughput*, *response time*, dan CPU *usage*. Adapun kriteria *throughput* didapatkan dari variabel ***NET I/O***, dan kriteria *response time* didapatkan dari variabel ***request rate*** pada hasil uji coba httperf. Sedangkan CPU *usage* dihasilkan dari rata – rata kolom ***idle*** dengan *tool sar*.

1) Pengujian *(Benchmark)* terhadap Parameter *Throughput*

Hasil pengujian performa web server *(benchmark)* terhadap parameter *throughput* dapat disajikan dalam Tabel 3.

Tabel 3.
Hasil Perbandingan Parameter *Throughput*

No.	Beban		Sebelum Konfigurasi	Setelah Konfigurasi
	Jumlah Koneksi	Konkurensi	Throughput (KB/s)	
1	10000	100	90.2	1805.5
	10000	200	180.5	3610.9
	10000	300	270.7	5416.1
	10000	400	360.9	7218.9
	10000	500	451.2	9012.7
2	20000	100	90.2	1805.5
	20000	200	180.5	3608.6
	20000	300	270.5	5099.2
	20000	400	360.9	6632.2
	20000	500	385.7	7597.8
3	30000	100	90.2	1805.5
	30000	200	180.5	3608.8
	30000	300	270.7	5389.4
	30000	400	303.1	6891.6
	30000	500	282.5	6650.3
4	40000	100	90.2	1805.5
	40000	200	180.5	3604.5
	40000	300	259.6	4609.1
	40000	400	283.7	5365.8
	40000	500	320.5	6103.8
5	50000	100	90.2	1805.5
	50000	200	180.5	3605.1
	50000	300	266.7	4811.5
	50000	400	269.2	5014.8
	50000	500	357.6	5546.9
	Rata-Rata		**242.68**	**4737.02**

Tabel 3 menunjukkan perbandingan nilai kriteria *throughput* yang dihasilkan oleh server sebelum konfigurasi dan server setelah konfigurasi. Adapun server sebelum konfigurasi menghasilkan nilai *throughput* sebesar 242.68 KB/s, sedangkan server sesudah konfigurasi menghasilkan nilai *throughput* sebesar 4737,02 KB/s. Selisih dari kedua hasil *throughput* tersebut sebesar **4494.32 KB/s.**

2) Pengujian *(Benchmark)* terhadap Parameter *Response Time*

Hasil pengujian performa web server *(benchmark)* terhadap parameter *response time* dapat disajikan dalam Tabel 4.
Tabel 4 menunjukkan hasil parameter *response time* pada saat pengujian atau *benchmark* terhadap server sebelum konfigurasi sebesar 288.876 *requests*/detik, sedangkan pada server setelah konfigurasi mendapatkan hasil kriteria *response time* sebesar 266.416 *requests*/detik. Hasil uji coba tersebut mendapatkan hasil bahwa server sesudah konfigurasi

menghasilkan nilai *response time* yang lebih kecil daripada server sebelum konfigurasi dengan selisih nilai sebesar **14.46 *requests*/detik.**

Tabel 4.
Hasil Perbandingan Parameter *Response Time*

No.	Beban		Sebelum Konfigurasi	Setelah Konfigurasi
	Jumlah Koneksi	Konkurensi	Response Time (Req/s)	
	10000	100	100	100
	10000	200	200	200
1	10000	300	300	300
	10000	400	400	399.8
	10000	500	500	499.2
	20000	100	100	100
	20000	200	200	199.9
2	20000	300	300	288.5
	20000	400	400	367.4
	20000	500	478	433.9
	30000	100	100	100
	30000	200	200	199.9
	30000	300	300	298.6
3	30000	400	348.5	382.4
	30000	500	375	400.5
	40000	100	100	100
	40000	200	200	199.7
	40000	300	291	262.7
4	40000	400	331.4	306.7
	40000	500	384.7	343.8
	50000	100	100	100
	50000	200	200	199.7
	50000	300	296.6	273.9
5	50000	400	348	285.3
	50000	500	468.7	318.5
	Rata-Rata		**280.88**	**266.42**

3) Pengujian *(Benchmark)* terhadap Parameter *CPU Usage*

Hasil pengujian performa web server *(benchmark)* terhadap parameter CPU *usage* dapat disajikan dalam Tabel 5. Tabel 5 menujukkan hasil pengujian performa web server atau *benchmark* mendapatkan hasil parameter CPU *usage* pada server sebelum konfigurasi dengan konsumsi CPU sebesar 77.86%, sedangkan konsumsi CPU yang dihasilkan pada server setelah konfigurasi sebesar 90.87%. Nilai penggunaan CPU diperoleh dari nilai seluruh sumber daya CPU yang ada yaitu 100% dikurangi hasil pengujian *benchmark* terhadap CPU.

a. Server sebelum konfigurasi:

$$100\% - 77.86 = 22.14\%$$

b. Server sesudah konfigurasi:

$$100\% - 90.87\% = 9.13\%$$

Jadi, hasil uji coba menunjukkan bahwa konsumsi CPU server sesudah konfigurasi lebih sedikit daripada server sebelum konfigurasi dengan selisih **13.01%**.

Dari ketiga parameter yang telah diuji coba, maka dapat diambil hasil bahwa server setelah konfigurasi lebih baik daripada server sebelum konfigurasi. Adapun parameter *throughput* menghasilkan server setelah konfigurasi lebih tinggi daripada server sebelum konfigurasi dengan selisih 4494.34 KB/s. Hal ini dikarenakan dalam prosesnya tejadi penanganan permintaan yang lebih banyak selesai tertangani seiring dengan durasi permintaan yang lebih cepat dan menghasilkan lintasan data yang lebih banyak.

Adapun parameter *response time* menghasilkan bahwa server setelah konfigurasi lebih cepat daripada server sebelum konfigurasi dengan selisih sebesar 14.46 *requests*/detik. Artinya web server setelah konfigurasi lebih cepat menangani permintaan *client* sebesar 14.46 *requests*/detik daripada wb server sebelum konfigurasi.

Adapun parameter CPU usage menghasilkan bahwa server setelah konfigurasi lebih sedikit mengkonsumsi CPU server daripada server sebelum konfigurasi dengan selisih sebesar 13.01%. Penggunaan CPU mewakili kinerja web server secara keseluruhan, semakin besar penggunaan CPU pada server berarti semakin lemah performa CPU yang dihasilkan.

Pada masing-masing kriteria baik *throughput*, *response time*, dan CPU *usage* menghasilkan bahwa server setelah konfigurasi memiliki nilai yang lebih unggul daripada server sebelum konfigurasi menurut masing – masing kriteria. Hal ini dikarenakan metode *clustering* dengan menggunakan konsep *high availabilit*y sangat mendukung kinerja server dengan cara membagi rata beban kerja baik pada *primary-server* dan *secondary-server* sehingga layanan bisa dilayani secara optimal tanpa menyebabkan server *down*. Selain itu virtualisasi berbasis kontainer juga bersifat *lightweight* atau ringan sehingga konsumsi pemakaian *resource* server lebih ringan daripada virtualisasi berbasis mesin virtual.

4. Kesimpulan

Kolaborasi virtualisasi dan *autoscaling* yang diterapkan pada layanan web server berbasis kontainer menggunakan PVE untuk orkestrasi mesin virtual dan kubernetes untuk orkestrasi kontainer berhasil diterapkan dengan website berbasis *CMS* yaitu wordpress dan *database* mysql. Pengujian *autoscaling* dilakukan dengan *benchmark* yaitu membebani web server dengan jumlah koneksi dan tingkat konkurensi yang sudah ditentukan, dan parameter pengujiannya menggunakan parameter *throughput*, *response time*, dan CPU *usage*. Adapun hasil pengujian membuktikan bahwa server setelah konfigurasi yaitu server yang menerapkan virtualisasi dan *autoscaling* lebih baik daripada server yang sebelum konfigurasi yaitu server yang tidak menerapkan virtualisasi dan *autoscaling* dengan perbandingan parameter *throughput* sebesar 4494.32 KB/s dan parameter *response time* sebesar 14.46 *Request*/detik serta parameter CPU *usage* sebesar 13.01%. Sehingga penelitian ini layak diterapkan di lingkungan Universitas Darussalam Gontor karena sesuai dengan tujuan yang diharapkan.

Pada penelitian ini belum memanfaatkan fitur Proxmox *VE* seperti *live migration*, selain itu belum ada *tool* untuk

monitoring layanan server secara berkala sehinnga diharapkan pada penelitian selanjutnya memanfaatkan fitur *live migration* pada Proxmox *VE* dan menggunakan *tool* monitoring seperti prometheus.

Tabel 5.
Hasil Perbandingan Parameter CPU *Usage*

No.	Beban		Sebelum Konfigurasi	Setelah Konfigurasi
	Jumlah Koneksi	Konkurensi	CPU Usage (%)	
	10000	100	78.76	91.35
	10000	200	79.22	91.84
1	10000	300	77.08	91.99
	10000	400	77.97	91.87
	10000	500	76.95	91.54
	20000	100	78.83	91.15
	20000	200	78.42	91.33
2	20000	300	77.73	91.39
	20000	400	77.49	91.46
	20000	500	77.69	91.45
	30000	100	78.94	91.67
	30000	200	78.48	91.37
	30000	300	77.19	91.34
3	30000	400	78.57	91.08
	30000	500	76.4	90.29
	40000	100	78.32	90.55
	40000	200	77.6	90.12
	40000	300	77.12	89.69
4	40000	400	76.28	90.4
	40000	500	76.77	90.31
	50000	100	76.77	89.58
	50000	200	78.83	90.15
	50000	300	77.71	89.47
5	50000	400	77.63	90.01
	50000	500	77.92	90.31
Rata-Rata			**77.86**	**90.87**

Referensi

[1] H. Hartono, *Pengertian Website dan Fungsinya,* Edisi 1, Ilmu Teknologi Informasi, Pematang, 2014.

[2] [2]Arfriandi, Perancangan, Implementasi, dan Analisis Kinerja Virtualisasi Menggunakan Proxmox, Vmware ESX,dan Openstack, *Jurnal Teknologi*, vol. 5 nomor 2, 2012, hal. 182–191.

[3] A. B. Bondi, *Characteristics of Scalability and Their Impact on Performance*. Ottawa: WOSP, 2004.

[4] C. Fiolution, *Modul Mastering Proxmox Cluster Storage*, no. 80.

[5] H. Saito, H. C. Lee, and C.-Y. Wu, *DevOps with Kubernetes*, Edisi 1. BIrmingham: Packt Publishing Ltd., 2017.

[6] E. A. Didik Sudayana, Virtualisasi Server Dengan Proxmox Untuk Pengoptimalisasian Penggunaan Resource Server Pada UPT Teknologi dan Komunikasi Pendidikan, *SATIN - Sains dan Teknologi. Inf.*, vol. 3, 2014, hal. 1-8.

[7] M. Tighe and M. Bauer, Integrating Cloud Application Autoscaling with Dynamic VM Allocation, *IEEE/IFIP NOMS 2014 - IEEE/IFIP Netw. Oper. Manag. Symp. Manag. a Softw. Defin. World*, 2014, hal. 1–9,.

[8] W. Supedana and D. P. Hostiadi, *High Availability Web Server Berbasis Open Source*, STMIK AMIKOM Yogyakarta, 2015, hal. 6–8, 2015.

[9] Y. T. Sumbogo, M. Data, and R. A. Siregar, Implementasi Failover Dan Autoscaling Kontainer Web Server Nginx Pada Docker Menggunakan Kubernetes, *J. Pengemb. Teknologi. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 12, 2018, hal. 6849–6854.

[10] W. Delnat, E. Truyen, A. Rafique, D. Van Landuyt, and W. Joosen, K8-Scalar: a Workbench to Compare Autoscalers for Container-Orchestrated Database Clusters, *13th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. (SEAMS 2018)*, 2018, hal. 33–39.

[11] A. Supriyadi and D. Gartina, Memilih Topologi Jaringan dan Hardware dalam Desain Sebuah Jaringan Komputer, *Inform. Pertanian*, vol. 16, no. 2, 2007, hal. 1037–1053.

[12] E. Casalicchio and V. Perciballi, Auto-Scaling of Containers: The Impact of Relative and Absolute Metrics, *Proc. - 2017 IEEE 2nd Int. Work. Found. Appl. Self* Syst. FAS*W 2017*, hal. 207–214.

[13] S. S. Hermawan and R. R. Saedudin, "Design of Cooling and Air Flow System Using NDLC Method Based on TIA-942 Standards in Data Center at CV Media Smart Semarang," *Int. J. Adv. Data Inf. Syst.*, vol. 1, no. 1, pp. 34–39, Apr. 2020.