Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 7, No. 7, Juli 2023, hlm. 3163-3167 http://j-ptiik.ub.ac.id

Implementasi Grafana untuk Auto-Scaling pada NGINX Web-Server di Lingkungan Docker Container

Rizki Tian Darmawan

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: dr_tian@student.ub.ac.id

Abstrak

Cloud Computing merupakan bidang teknologi yang menyediakan berbagai layanan melalui internet, berdasarkan teknologi virtualisasi seperti virtualisasi penyimpanan, jaringan, dan komputasi. Virtualisasi komputasi melibatkan penggunaan mesin virtual (VM) dan kontainer. VM memiliki overhead yang besar, sementara kontainer menawarkan performa lebih baik karena overhead yang lebih kecil. Teknologi kontainer memberikan banyak manfaat, termasuk lingkungan konsisten dan reproduksi sumber daya, serta kemudahan dalam deployment. Auto-scaling membantu kontainer memungkinkan skala sumber daya beradaptasi dengan permintaan, memastikan layanan berjalan tanpa kegagalan fungsionalitas, dan mencegah pemborosan sumber daya saat permintaan rendah. Grafana yang merupakan platform untuk visualisasi dan alerting dari data yang terintegrasi ke aplikasi memungkinkan untuk pemantauan dan notifikasi kepada aplikasi lain berdasarkan anomalitas penggunaan CPU rerata dari kontainer yang kurang atau lebih dari batas normal yang ditentukan. Dengan dilakukannya uji coba implementasi Auto-scaling pada docker container pada Grafana, maka didapatkan pengujian behasil dengan aktivasi alarm Grafana pada pergerakan rerata CPU di atas 70% dan melakukan scale in dalam aktivitas rerata penggunaan CPU container diawah 20%.

Kata kunci: *kontainer, auto-scaling, scale-in, scale-out*

Abstract

Cloud Computing is a technological field that provides various services over the internet, based on virtualization technologies such as storage virtualization, networking, and computing. Computational virtualization involves the use of virtual machines (VMs) and containers. VMs have significant overhead, while containers offer better performance due to lower overhead. Container technology offers numerous benefits, including consistent and reproducible resource environments, as well as ease of deployment. Auto-scaling aids containers in dynamically adjusting resource scaling according to demand, ensuring uninterrupted service functionality and preventing resource wastage during low-demand periods. Grafana, a platform for visualization and data-integrated alerting within applications, enables monitoring and notifications to other applications based on anomalies in average CPU usage from containers that are above or below the specified normal threshold. By conducting implementation tests of Auto-scaling on Docker containers within Grafana, successful testing is achieved through Grafana alarm activation in cases of average CPU movement exceeding 70%, and performing scale-in activities when the average CPU usage of the container falls below 20%.

Keywords: c*ontainer, auto-scaling, scale-in, scale-out*

1. PENDAHULUAN

Cloud Computing adalah bidang teknologi yang meliputi informasi teknologi yang menawarkan berbagai layanan seperti komputasi, penyimpanan, dan jaringan melalui internet. Layanan yang ditawakan didasari oleh teknoogi virtualisasi. Seperti virtualisasi penyimpanan, jaringan dan komputasi. Virtualisasi komputasi yang digunakan pada cloud computing mengacu pada abstraksi perangkat keras menjadi. Komputasi berbasis virtual meliputi *virtual machine* dan *container*. VM memiliki overhead yang besar sehingga mempengaruhi performa, dengan adanya dampak tersebut, kontainer menjadi pilihan

karena kontainer hanya akan menjalankan aplikasi dan beberapa tools untuk mendukung aplikasi tersebut, Kontainer sendiri adalah lightweight platform yang digunakan untuk migrasi data yang dapat berjalan secara efisien. Pada kontainer program di kemas dengan library-nya, konfigurasi file, dan semua komponen yang dibutuhkan untuk program itu berjalan. Layaknya peti kemas, jadi aplikasi dan konfigurasinya akan terisolasi terhadap aplikasi atau kontainer lain yang berjalan seiringan pada sistem yang sama (Xavier et al., 2013). Penggunaan kontainer sebagai teknologi dasar untuk menerapkan aplikasi berskala besar membuka banyak tantangan dalam bidang manajemen sumber daya saat berjalan. Salah satu karateristik kontainer adalah aplikasi deployment bisa dengan mudah diluncurkan. Sehingga menyediakan lingkungan yang konsisten dan dapat direproduksi. Salah-satu Teknik yang dapat digunakan untuk mereproduksi sumberdaya kontainer adalah auto-scaling. Auto scaling pada container mengizinkan pengguna untuk mengotomasi skala sumberdaya secara dinamis berdasarkan banyaknya permintaan (Mao & Humphrey, 2012). Pada saat permintaan melonjak (peak traffic), jumlah kontainer akan diskalakan sehingga server dapat melayani semua permintaan tanpa mengalami kegagalan fungsionalitas. Sebaliknya pada saat permintaan rendah, jumlah kontainer dapat diturunkan sehingga tidak ada sumberdaya yang terbuang. Auto-Scaling menjamin pendukung infrastruktur dari aplikasi dapat beradaptasi secara dinamis terhadap permintaan yang meningkat atau menurun secara on-demand.

Grafana adalah aplikasi platform *open-source* monitoring yang digunakan untuk manajemen visualisasi dan analisa data dari berbagai sumber secara *real-time* (Grafana, 2015). Platform ini memiliki fitur-fitur utama yang diperlukan untuk *auto-scaling* seperti Integrasi Sumber Data, Visualisasi, Alerting. Visualisasi dari sumber data yang telah terintegrasi ke Grafana. Alerting dapat mengamati dan mengambil perintah berdasarkan anomalitas dari data yang dipantau dan membuat permintaan ke layanan lain seperti Webhook untuk notifikasi dan eksekusi lanjut. Grafana digunakan pada penelitian ini karena jika dibandingkan dengan implementasi *Alerting* yang diterapkan oleh Alert Manager yang ada pada Prometheus, meskipun memiliki fungsi yang sama yakni mengamati value dari penggunaan sumberdaya dari subjek yang ditentukan Grafana yang memiliki visualisasi yang dapat dipresentasikan memiliki keunggulan di sini.

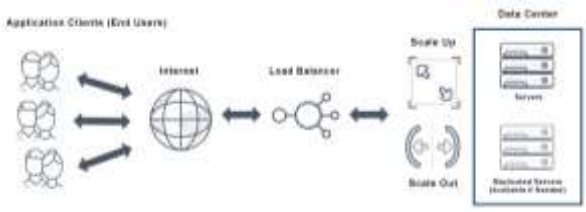
Pada penelitian yang berjudul Analisis Performansi Container Docker, LXC dan LXD Pada Web Server, FTP Server dan Mail Server menguji performa dari berbagai platform kontainer Docker, LXC dan LXD kemudian menunjukan bahwa Container Docker yang digunakan sebagai web server lebih unggul dibandingkan platorm LX, dan LXD dengan delay masing-masing 8,9,9. Kemudian pada penelitian A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments menunjukkan bahwa auto-scaling pada kontainer diperlukan terutama pada aplikasi yang diperuntukkan berjalan secara konsisten. Setelah yang pemaparan diatas, auto scaling dibutuhkan untuk faktor-faktor memelihara aplikasi web yang terkontainersasi seperti availability, performa, skalabilitas dari aplikasi yang terkontainerisasi. Dimana itu mengizinkan efisiensi alokasi sumber-daya dan menjamin pengalaman penggunaan yang seamless. Untuk itu “auto-scaling terhadap docker container” diperlukan.

2. KONTAINER

Kontainer adalah sebuat *lightweight platform* yang digunakan untuk migrasi data yang dapat berjalan secara efisien. Alasan kenapa aplikasi dapat berjalan diatas kontainer. Pada kontainer program, aplikasi dikemas dengan library-nya, konfigurasi file, dan semua komponen yang dibutuhkan untuk program itu berjalan (Xavier et al., 2013). Sehingga menyediakan lingkungan yang konsisten dan dapat direproduksi.

3. AUTO-SCALING

Auto-Scaling adalah kemampuan untuk berskalasi secara otomatis. *Auto-scaling* pada *cloud computing* adalah fitur yang mengizinkan pengguna untuk me-manajemen aplikasi secara dinamis mengatus jumlah sumberdaya komputatif yang dilakokasikan terhadap aplikasi berbasis penggunaan daya (Mao & Humphrey, 2012).



Gambar 1. Diagram *auto scaling*

Auto Scaling pada Kontainer dapat dibedakan menjadi 2 tipe, yakni Auto Scaling Horizontal dan *Auto Scaling* Vertical (Rossi et al., 2019). Untuk *Auto Scaling* Horizontal adalah kemampuan untuk meningkatkan kapasitas dengan menghubungkan beberapa *hardware* ataupun *software* untuk dapat bekerja sebagai satu sistem yang sama.

4. DOCKER

Docker adalah sebuah *platform open-source* untuk mengembangkan, mengirim dan menjalankan aplikasi dalam *container*. Docker membantu pengembang menjalankan aplikasi yang konsisten dan *portable* untuk berbagai *platform* kerja yang tersedia seperti, *desktop*, *server* dan *cloud* (Docker Team, 2013).

Docker container aplikasi dan semua dependansinya terkemas dalam satu paket dan mudah di distribusikan dalam sebuah image yang dapat di unggah ke *docker registry* serupa untuk di pull oleh pengguna lain.

5. DOCKER

Docker adalah sebuah platform open-source untuk mengembangkan, mengirim dan menjalankan aplikasi dalam container. Docker membantu pengembang menjalankan aplikasi yang konsisten dan portable untuk berbagai platform kerja yang tersedia seperti, desktop, server dan cloud (Docker Team, 2013).

6. PROMETHEUS

Prometheus adalah aplikasi yang membantu memonitor tiap aplikasi dan node yang mengekspor data metriknya ke prometheus untuk dapat disajikan oleh pengguna dalam bentuk http (Prometheus Team, 2014).

Prometheus bertugas untuk mengkoleksi tiap data metrik dari target yang mengekspor ke aplikasi, metrik tersebut adalah data yang telah ditetapkan sesuai keadaan di saat itu, kemudian mengevaluasi aturan ekspresi, dan menampilkan hasilnya, dan mendapatkan feedback yang sesuai jika kondisi diamati dengan benar.

7. GRAFANA

Grafana adalah aplikasi platform open-source monitoring yang digunakan untuk manajemen visualisasi dan analisa data dari berbagai sumber secara real-time. Fitur utama yang terlibat untuk penelitian kali ini adalah :

- Integrasi Sumber Data: Grafana dapat menghubungkan ke berbagai sumber data, membuat Grafana menjadi tools yang serbaguna dan mudah beradaptasi untuk berbagai lingkungan data yang ada
- Visualisasi: Grafana menyediakan berbagai opsi visualisasi yang dibutuhkan pengguna untuk menyajikan data sesuai kebutuhan secara efektif. Tipe visualisasi pun beragam seperti diagram, grafis, gauge meter dan masih banyak lagi.
- Alerting: Platform Grafana menyediakan pengguna untuk menyetel alarm berdasarkan kondisi yang ditentukan, membantu untuk mengidentifikasi dan merespon kepada kejadian yang krusial atau kejanggalan pada data.

Dalam penelitian ini Grafana sangat membantu untuk implementasi auto-scaling. Fitur yang terlibat pada auto-scaling adalah Visualisasi dan Alerting dari data penggunaan sumberdaya yang ditampilkan. Alerting dapat mengamati dan mengambil perintah berdasarkan anomalitas dari data yang dipantau dan membuat permintaan ke layanan lain untuk notifikasi.

8. LOAD-BALANCING

Load balancing adalah Teknik yang digunakan pada jaringan computer dan sistem yang terdistribusi untuk mengefisiensikan jaringan traffik yang masuk terhadap beberapa server, menjamin pemerataan sumberdaya yang optimal, meningkatkan performa, dan high availability (Membrey, et al., 2012). Bertujian untuk mencegah salah satu server bekerja lebih banyak daripada yang diperlukan dengan menjamin alokasi kerja di distribusikan lebih merata kepada setiap server yang tersedia.

Cara load balancing bekerja yakni bertindak sebagai penengah antara server dan klien. Ketika klien mendistribusikan permintaan kepada layanan atau sumberdaya tertentu seperti website contohnya, load balancer menerima permintaan dan menentukan server terbaik untuk menerima pada saat itu. Load balancer menggunakan rangkaian algoritma yang tersedia atau aturan untuk menentukan aksi berikut. Kemudian

meneruskan permintaan kepada server yang telah ditentukan sebelumnya dan mengembalikan respon dari server tersebut kembali ke klien.

9. CADVISOR

cAdvisor, yang dapat juga dikenal sebagai Container Advisor, menyediakan pengguna kontainer pengertian secara menyeluruh dari bagaimana container yang berjalan, dan memanfaatkan sumberdaya yang ada (Porter, et al., 2014). cAdvisor berfungsi sebagai proses yang berjalan secara terus menerus yang mengkoleksi, mengabungkan, analisa, dan mengirim informasi yang berhubungan dengan kontainer yang berjalan. Secara spesifik menyimpan parameter sumberdaya yang terisolasi, penggunaan sumberdaya secara historis, dan rekaman secara detil dari penggunaan sumberdaya secara waktu waktu, dan rekaman jaringan statistika tiap kontainer. Data ini tersedia untuk dirikim pada tingkat kontainer dan mesin yang berjalan.

cAdvisor didesain untuk bekerja secara baik dengan kontainer Docker dan secara sengaja di desain dapat pula bekerja dengan tipe kontainer lain tanpa membutuhkan konfigurasi tambahan

10. PERANCANGAN

Perancangan menjelaskan bagaimana untuk membangun metode Auto-Scaling pada Docker *container*.

Perancangan ditunjukkan memudahkan sistem implementasi. Tahap perancangan adalah tahapan untuk mempersiapkan impementasi Kebutuhan Fungsional. Berikut adalah rancangan implementasi sistem:

Gambar 3.1. Perancangan sistem

Berdasarkan gambar diatas akan dilakukan Perancangan (poin satu) sebelum memulai implementasi. Perancangan meliputi instalasi tools yang dibutukan untuk menjalankan auto-scaling pada aplikasi yang terkontainerisasi. Docker dan aplikasi kontainer tidak di ikutsertakan dikarenakan implementasi ini dijalankan ketika docker dan aplikasi yang akan dieksekusi telah berjalan. Setelah tools di instalasi. Metrik dari kontainer web server yang diperoleh oleh cAdvisor akan di berikan oleh Prometheus yang akan dikoleksikan oleh visualisasikan oleh Grafana. Kemudian setelah proses persiapan perancangan akan dilanjutkan ke proses auto scaling itu sendiri.

Proses akan berlanjut pada Auto Scaling kontainer. Proses ini meliputi cara eksekusi program dan tools-tools yang telah terinstall. Diawali oleh Grafana yang memonitor penggunaan metrik seluruh kontainer web dan melihat metrik dari penggunaan CPU dari seluruh kontainer (poin 2). Jika penggunaan CPU dibawah 10% maka Grafana akan mengirimkan alert yang memerintahkan untuk mengeksekusi Scale in script (poin 3). Scale in Script berisi perintah untuk menghapus replika kontainer nginx jika melebihi jumlah replika awal. Dan Jikalau penggunaan CPU diatas 50% maka Grafana akan mengirim alert yang memerintahkan untuk mengeksekusi Scale out script (poin 4). Scale out script berisi untuk menambahkan replika dari kontainer yang ada sehingga membantu agar replika yang sedang berjalan tidak beraktifitas lebih berat. Setelah script Scale out atau Scale in dijalankan maka akan dilakukan pembaharuan configurasi dari load balancing untuk dapat memasukkan atau menghapus replika terbaru. Tetapi jikalau penggunaan CPU berada di antara 10% dan 50% maka Grafana akan tetap berjalan seperti biasa.

11. HASIL DAN PEMBAHASAN

Dari implementasi auto-scaling pada nginx container didapatkan hasil waktu sebagai berikut:

Pengujian Scale-Out

Peng-guna	Kondisi Awal		Kondisi Akhir		
	Replika	CPU	Replika	CPU	
100	3	45%	3	45%	X
150	3	60%	3	60%	X
200	3	70%	4	45%	✓
300	3	70%	5	50%	✓
400	3	70%	6	50%	✓
500	3	70%	3	60%	✓
600	3	70%	3	60%	✓

Pengujian Scale-In

Peng-	Kondisi Awal	Kondisi Akhir	

guna	Replika	CPU	Replika	CPU	
0	10	0%	3	0%	✓
0	9	0%	3	0%	✓
0	8	0%	3	0%	✓
0	7	0%	3	0%	✓
0	6	0%	3	0%	✓
0	5	0%	3	0%	✓
0	4	0%	3	0%	✓
0	3	0%	3	0%	X

Dari kedua hasil pengujian Skenario Scale-Out dan Scale-In sini dapat diambil kesimpulan juga Grafana mampu untuk menjalankan Auto-Scaling skenario Scale-Out ketika penggunaan Rerata CPU menyentuh 70% sesuai ambang batas atas yang di tentukan sebelumnya pada Grafana. Untuk skenario scale-in Grafana mampu untuk menangkap penggunaan rerata CPU meskipun penggunaan rerata CPU dari awal adalah 0% dikarenakan sudah sejak awal memasuki batas untuk eksekusi scale-in.

12. KESIMPULAN

Berdasarkan hasil perancangan dan implementasi dari Auto-Scaling pada Docker container didapat kesimpulan berupa:

Untuk merancang Auto-Scaling terhadap Docker Container menggunakan pendekatan auto-scaling horizontal dikarenakan untuk menerapkan pendekatan auto-scaling vertikal dengan merubah spesifikasi pada docker kontainer membutuhkan penghapusan kontainer sebelumnya untuk menambahkan kontainer baru dengan spesifikasi yang berbeda dengan kontainer sebelumnya, yang cara ini dinilai kurang efektif dibandingkan penerapan horizontal yang hanya menyesuaikan jumlah replika container dengan spesifikasi yang sama.

Grafana digunakan pada penelitian ini karena jika dibandingkan dengan implementasi Alerting yang diterapkan oleh Alert Manager yang ada pada Prometheus, meskipun memiliki fungsi yang sama yakni mengamati value dari penggunaan sumberdaya dari subjek yang ditentukan Grafana yang memiliki visualisasi yang dapat dipresentasikan memiliki keunggulan di sini.

Hasil dari implementasi Auto-Scaling pada Docker container didapatkan kesimpulan didapati dengan adanya auto-scaling maka untuk skalasi container ketika ada traffic yang tidak menentu dapat dijalankan secara otomatis dengan konfigurasi scale out jika batas 70% dan scale in dengan batas 20% sesuai konfigurasi dari Grafana.

13. DAFTAR PUSTAKA

Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. Journal of Grid Computing, 12(4), 559–592. https://doi.org/10.1007/s10723-014-9314-7

Mao, M., & Humphrey, M. (2012). Cloud Auto-Scaling with Deadline and Budget Constraints. https://doi.org/https://doi.org/10.18130/V31B83

Membrey, P., Hows, D., & Plugge, E. (2014). Load Balancing Basics. In Practical Load Balancing (Vol. 7, pp. 109–116). https://doi.org/https://doi.org/10.1007/978-1-4302-3681-8_7

Rossi, F., Nardelli, M., & Cardellini, V. (2019). Horizontal and vertical scaling of container-based applications using reinforcement learning. IEEE International Conference on Cloud Computing, CLOUD, 2019-July, 329–338. https://doi.org/10.1109/CLOUD.2019.00061

Wahanani, H. E., Idhom, M., Faris, M., & Saputra, E. (n.d.). Seminar Nasional Informatika Bela Negara (SANTIKA) Analisis Performansi Container Docker, LXC dan LXD Pada Web Server, FTP Server dan Mail Server. 1(2020).

Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., & De Rose, C. A. F. (2013). Performance evaluation of container-based virtualization for high performance computing environments. Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013, 233–240. https://doi.org/10.1109/PDP.2013.41

Membrey, P., Hows, D. & Plugge, E., 2012. Practical Load Balancing. In: Load Balancing Basics. s.l.:Apress, pp. 109-116.