Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 1, Januari 2018, hlm. 11-20 http://j-ptiik.ub.ac.id

Implementasi Alokasi Memori Dinamis pada Sistem Komputer Berbasis *Clustering* K-Means

Florensius Tri Putra Simamora[1], Widhi Yahya[2], Sabriansyah Rizqika Akbar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]florensiusputra@gmail.com, [2]widhi.yahya@ub.ac.id, [3]sabrian@ub.ac.id

Abstrak

Virtualisasi adalah proses pembuatan bentuk dalam versi virtual dari sistem operasi, jaringan dan lain-lain yang bisa menjalankan aplikasi layaknya perangkat fisik. Sumber daya dalam virtualisasi menjadi faktor penting untuk mendukung kinerja *server*. Terkadang sumber daya tidak dimanfaatkan sepenuhnya bila alokasi sumber daya tetap. Akibatnya, ketika server membutuhkan lebih banyak sumber daya, alokasi memori yang tidak terpakai tidak dapat digunakan. Dengan Alokasi memori, sistem dapat mengatur penggunaan memori untuk menjaga dan meningkatkan kinerja server yang kadang-kadang dapat berubah. Penerapan alokasi memori dapat dimudahkan dengan adanya algoritma dan metode k-means regresi untuk melakukan prediksi terhadap penggunaan sumber daya memori. Untuk itu, penulis telah mengimplementasikan sebuah sistem alokasi memori dinamis pada sistem komputer berbasis *clustering* k-means. Dari hasil pengujian, sistem mampu melakukan perubahan berupa *upgrade* dan *downgrade* dengan bantuan *tools Apache Jmeter* dengan waktu 2.5 detik. Berdasarkan hasil pengujian yang sudah dilakukan, *hypervisor* berhasil melakukan *upgrade* pada detik ke 36.5699 dari 1 GB ke 2 GB. Kemudian detik ke 153.1348 *hypervisor* melakukan *upgrade* lagi dari 2 GB ke 4 GB. Setelah request ke *server* berhenti, maka *hypervisor* melakukan *downgrade* pada detik ke 611.709 dari 4 GB ke 2 GB kemudian *downgrade* lagi pada detik ke 624.4994 dari 2 GB ke 1 GB.

Kata Kunci: Virtualisasi, K-Means, Regresi Linier, Alokasi Memori

Abstract

Virtualization is the process of creating forms in virtual versions of operating systems, networks and others that can run applications like physical devices. Resources in virtualization become an important factor to support server performance. Sometimes resources are not fully utilized when resource allocation remains. As a result, when the server requires more resources, unused memory allocations can not be used. With memory allocation, the system can manage memory usage to maintain and improve server performance that can sometimes change. Application of memory allocation can be facilitated by algorithm and k-means regression method to predict the use of memory resources. To that end, the authors have implemented a dynamic memory allocation system on computer-based k-means clustering system. From the test results, the system is able to make changes in the form of upgrades and downgrades with the help of tools Apache Jmeter with time 2.5 seconds. Based on the results of tests that have been done, hypervisor successfully upgraded at 36.5699 seconds from 1 GB to 2 GB. Then second to 153.1348 hypervisor upgrade again from 2 GB to 4 GB. After the request to the server stops, the hypervisor downgrades the seconds to 611.709 from 4 GB to 2 GB then downgrade again in seconds to 624.4994 from 2 GB to 1 GB.

Keywords: *Virtualization, K-Means, Linier Regression, Memory Allocation*

1. PENDAHULUAN

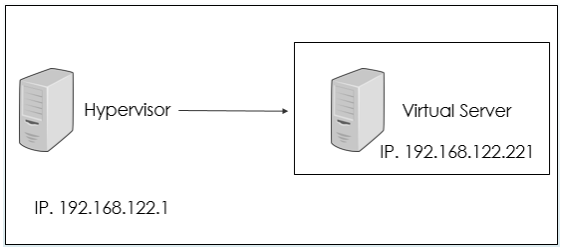
Virtualisasi adalah proses pembuatan bentuk dalam versi virtual dari sistem operasi, jaringan dan lain-lain (Kaciak, Rajok, Gufron, 2013) yang bisa menjalankan aplikasi layaknya perangkat fisik. Teknik virtualisasi digunakan untuk mengurangi jumlah *server* dalam bentuk fisik. Untuk mengatur dan mengawasi mesin virtual, virtualisasi menggunakan perangkat lunak yaitu *hypervisor*. *Hypervisor* merupakan suatu teknologi virtualisasi yang dikembangkan Linux. KVM *hypervisor* memungkinkan

seseorang dapat menjalankan satu atau lebih sistem operasi dalam Linux maupun Windows (Soriga & M., Barbulescu. 2013). Teknologi virtualisasi memungkinkan beberapa VPS (*Virtual Private Server*) jalan dalam satu *server* fisik saja. *Server* adalah salah satu bagian dari jaringan komputer yang memegang peran penting dalam memberikan layanan kepada pengguna. *Server* harus bisa memenuhi permintaan pengguna agar bisa mendukung kebutuhan kinerja pengguna. Oleh sebab itu, ketersediaan memori yang tinggi menjadi syarat untuk tersedianya layanan bagi pengguna (Niswar, Sabri, Warni, Musa. 2013).

Penggunaan sumber daya memori (ram) di virtualisasi *server* terkadang tidak dimanfaatkan secara efisien bila alokasi sumber dayanya tetap. Akibatnya, ketika *server* membutuhkan sumber daya lebih, maka alokasi memori yang tidak terpakai tidak dapat digunakan (Niswar, Sabri, Warni, Musa. 2013). Sistem operasi memiliki tugas yang besar karena bertindak sebagai antarmuka antara sumber daya utama seperti perangkat keras dan aplikasi yang berjalan. Sistem operasi harus mampu melakukan alokasi memori yang suatu waktu memori *server* dapat berubah (Awais, Muhammad Abdullah. 2016). Oleh sebab itu, diperlukan sebuah metode untuk melakukan alokasi memori secara dinamis pada *server* berdasarakan proses *monitoring*. Pada penelitian sebelumnya (Sendy Deva Pastia. 2015) tentang Implementasi Purwarupa Sistem Manajemen Sumber Daya Dinamis pada Virtual *Server* menggunakan metode Regresi Linier sebagai proses pengalokasian memori. Alokasi memori dilakukan dengan proses *monitoring* memori, prediksi, rekomendasi, dan rekonfigurasi untuk mendapatkan hasil prediksi dalam memberikan prediksi pada *server* agar *server* tidak mengalami kekurangan memori yang dapat mengakibatkan *server* tidak bekerja. Proses alokasi memori atau rekonfigurasi dilakukan dengan mematikan *server* dahulu agar jumlah memori pada *server* bertambah. Dengan hal ini, *server* akan memiliki jumlah memori yang lebih dari sebelumnya. Berdasarkan jurnal (Niswar, Sabri, Warni, Musa. 2013) tentang *Memory Sharing Management on Virtual Private Server* dengan menggunakan teknik virtualisasi *OpenVZ*. *OpenVZ* adalah teknik virtualisasi untuk menjalankan banyak VPS dalam satu *server* fisik saja. Alokasi memori dilakukan dengan cara berbagi alokasi memori antar VPS. Alokasi memori didasarkan jika VPS yang satu membutuhkan memori, maka VPS lain akan meminjamkan memorinya ke VPS yang membutuhkan. Dengan cara ini, maka kinerja dari VPS dan utilitas RAM menjadi meningkat. Berdasarkan permasalahan yang telah diuraikan dan penelitian sebelumnya, penulis akan membangun sistem *monitoring*, rekomendasi dan rekonfigurasi secara otomatis tanpa mematikan *server* atau menggangu kinerja dari *server*. Proses *monitoring* dilakukan di *hypervisor* karena tugasnya adalah untuk mengawasi *server* yang suatu waktu berubah, rekomendasi dilakukan berdasarkan hasil monitoring dan rekonfigurasi dilakukan berdasarkan hasil rekomendasi. Proses pengalokasian memori menggunakan sebuah algoritma dan metode untuk proses prediksi, guna meningkatkan penggunaan virtual server. Algoritmanya adalah *clustering K-Means* sedangkan metodenya yaitu *Regresi Linier*. K-Means adalah sebuah algortima yang mengelompokkan data sama ke dalam satu kelompok dan mengelompokkan data yang berbeda ke kelompok yang lain untuk tujuan memaksimalkan jarak *intra-cluster* dan meminimalkan jarak *inter-cluster* (Ketu, Agarwal, 2015). Berdasarkan penggunaan memori, k-means sedemikian rupa mengatasi masalah dengan jumlah data banyak dengan tujuan k-means dapat meningkatkan kemampuan pengolahan. Regresi linier merupakan sebuah metode untuk menyelesaikan permasalahan, tentunya sangat berlaku dalam prediksi. Regresi linier dipergunakan untuk meramalkan apabila pola pada masa lampau dari data yang sebenarnya menunjukkan kecenderungan naik dari waktu ke waktu. Dengan adanya k-means regresi, dapat memperoleh penanganan yang lebih baik, dimana k-means mengelompokkan data dari hasil *monitoring* agar mudah diproses yang kemudian dibantu oleh regresi dalam proses prediksi untuk melihat kecenderungan penggunaan memori dari suatu waktu dengan tujuan mendapatkan prediksi dari hasil *monitoring* memori. Hasil dari *monitoring* memori akan diuji dengan menggunakan *tools Apache Jmeter* untuk memberikan beban terhadap *server* guna untuk melihat hasil *upgrade*, *downgrade* dan lama rekonfigurasi.

2. PERANCANGAN DAN IMPLEMENTASI

2.1. Perancangan Sistem

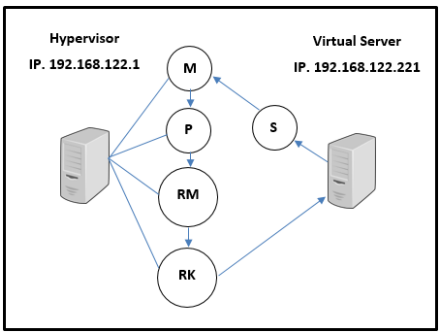


Gambar 1. Perancangan Sistem

Berdasarkan Gambar 1 perancangan sistem ini dijelaskan sebagai berikut.

1. Perancangan *hypervisor*, memuat proses untuk monitoring memori pada virtual *server*. *Hypervisor* juga mampu untuk memprediksikan hasil penggunaan ram yang kemudian dapat direkomendasikan dan direkonfigurasikan secara otomatis tanpa peran administrator.
2. Perancangan Virtual *server*, memuat proses untuk monitoring memori.

2.1.1 Komponen implementasi Sistem



Gambar 2. Komponen Implementasi Sistem

Penjelasan :

S : Start untuk virtual *server*
M : *Monitoring* di *hypervisor*
P : Prediksi di *hypervisor*
RM : Rekomendasi di *hypervisor*
RK : Rekonfigurasi di *hypervisor*

Pada gambar diatas, topologi perancangan sistem terdiri dari virtual *server* dan *hypervisor* dalam satu komputer fisik. Proses dimulai dari *Start* (S) sampai Rekonfigurasi (RK), setelah direkonfigurasi maka sistem selesai. Untuk memulai proses *monitoring* memori terhadap *server*, terlebih dahulu menjalankan *server* (S) dengan mengeksekusi sebuah script "*python monitor &"* untuk mendapatkan hasil *monitoring* memori yang kemudian *hypervisor* akan melakukan *monitoring* (M) memori secara realtime melalui terminal dengan script "*kmeans-regression.py".* Seiring berjalannya *monitoring* maka akan dilakukan prediksi terhadap memori.

Untuk memberikan beban pada *server*, harus menggunakan *tools apache jmeter*. *Apache jmeter* digunakan untuk memberikan request ke *server* agar memori *server* naik. Jika proses peramalan memori melampaui batas 80% maka *hypervisor* akan melakukan rekomendasi (RM) terhadap *server*. Setelah dilakukan rekomendasi, maka *hypervisor* akan melakukan rekonfigurasi (RK) untuk mengubah jumlah memori *server* sesuai dengan kebutuhan sistem.

2.1.2 Topologi

Gambar 3. Topologi Perancangan Sistem

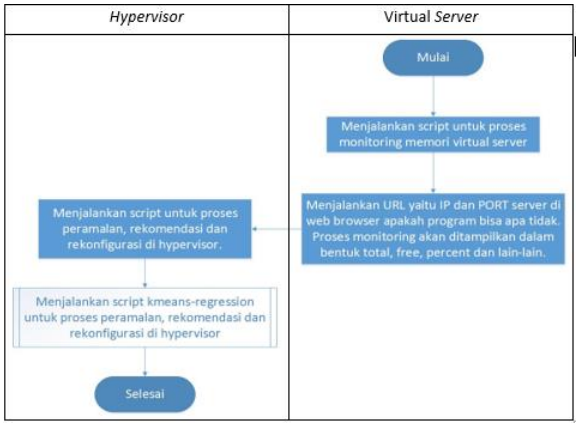
Berdasarkan Gambar 3 Komponen implementasi sistem ini dijelaskan sebagai berikut.

1. Tahap *monitoring* adalah proses untuk melihat perilaku dari memori .
2. Tahap prediksi dilakukan bersamaan dengan tahap *monitoring*. Prediksi bertujuan untuk meramalkan penggunan memori yang suatu waktu overload yang bisa merusak *server*. Pada proses *monitoring*, prediksi baru bisa diramalkan setelah sesuai dengan syarat yang ditentukan. Proses prediksi memiliki 2 syarat. Syarat pertama adalah jumlah cluster. Setelah proses *monitoring* berjalan mulai dari detik ke 0, proses prediksi belum bisa diramalkan, dikarenakan pada detik ke 0 sampai ke 5 masih pada proses penentuan atau pencarian k-means. K-Means di set sebanyak 5, jadi prediksi baru bisa diramalkan setelah k-means sudah terpenuhi. Syarat kedua adalah waktu. Sistem baru bisa meramalkan 60 detik setelah waktu berjalan. Sebagai contoh,

realtime detik ke 6 berarti prediksi detik ke 66.

3. Proses rekomendasi adalah untuk memberikan daya yang baru jika memori melebihi batas. Proses rekomendasi dilakukan setelah proses prediksi selesai dilakukan. Rekomendasi dilakukan jika melebihi batas 80% atau kurang dari 10%. Jika melebihi batas 80% atau kurang dari 10%, maka akan dijalankan proses rekomendasi memori yang kemudian akan direkonfigurasi sesuai dengan hasil rekomendasi ke virtual *server*. Setelah direkonfigurasi, maka *server* akan memiliki memori yang baru.

2.1.3 Diagram Alir Perancangan Sistem



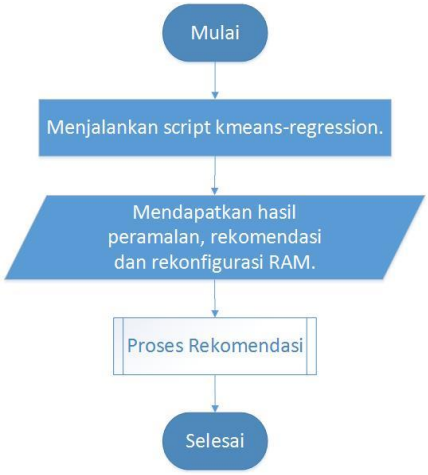
Gambar 4 Diagram alir *monitoring* sampai rekonfigurasi

Berdasarkan Gambar 4 diagram alir sistem ini terdiri dari 2 tahapan, yang dijelaskan sebagai berikut.

1. Proses pertama kali dimulai dari virtual *server* dengan menjalankan script untuk proses *monitoring* memori, setelah itu cek di web browser apakah program bisa atau tidak dengan menggunakan URL, IP dan PORT. URL ini akan kemudian akan dijadikan oleh *hypervisor* untuk *monitoring* memori dari virtual *server*.
2. Proses kedua adalah *hypervisor*, pada tahap ini *hypervisor* menjalankan script k-means-regresion untuk proses prediksi, rekomendasi dan rekonfigurasi.

2.1.4 Diagram Alir Prediksi

Berdasarkan Gambar 5 diagram alir prediksi, hal yang pertama dilakukan adalah menjalankan script k-means-regerssion di *hypervisor* untuk mendapatkan prediksi yang kemudian akan direkomendasi jika sistem memerlukan ruang tambahan. Rekomendasi dilakukan jika hasil prediksi melampaui batas 80%. Setelah direkomendasi, selanjutnya akan direkonfigurasi ke virtual *server* untuk perubahan memori sesuai dengan hasil rekomendasi.

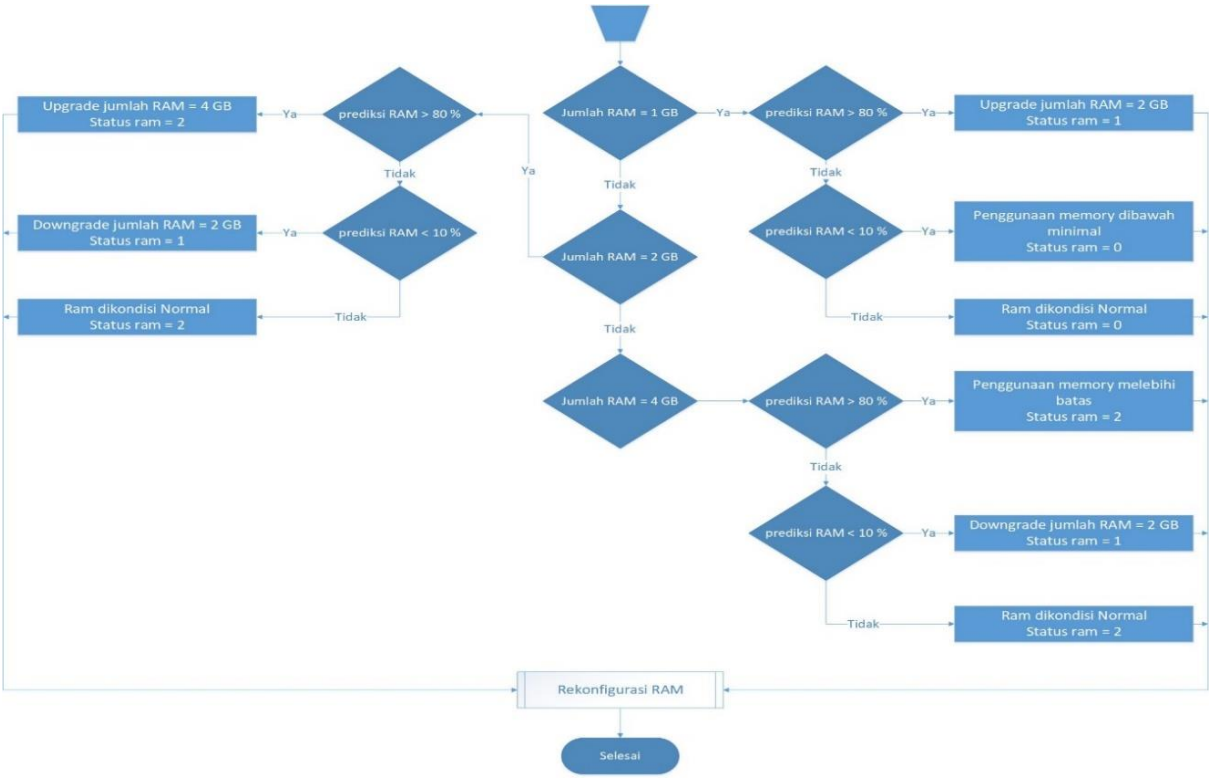


Gambar 5 Diagram Alir Prediksi

2.1.5 Diagram Alir Rekomendasi

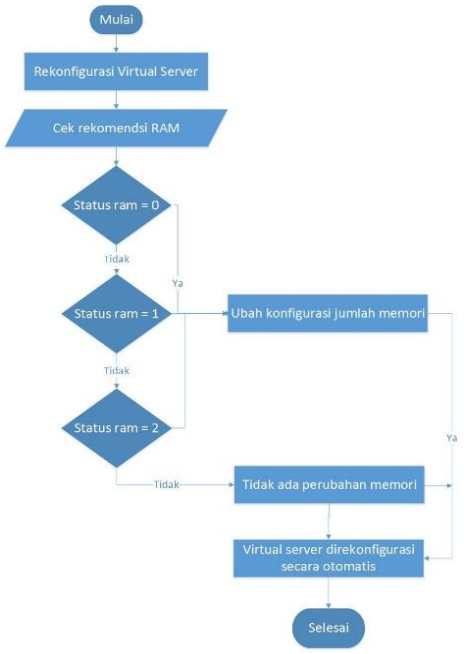
Berdasarkan Gambar 6 diagram alir rekomendasi dapat dijelaskan sebagai berikut.

1. Apabila kapasitas RAM adalah 1 GB dan prediksi lebih dari 80% maka rekomendasi adalah *upgrade* kapasitas RAM menjadi 2 GB. Konfigurasi langsung 2 GB ke virtual *server*. Status 1.
2. Apabila kapasitas RAM adalah 1 GB dan prediksi kurang dari 10% maka rekomendasi adalah Memunculkan peringatan “Penggunaan Memori dibawah Minimal. Status 0.
3. Apabila kapasitas RAM adalah 2 GB dan prediksi lebih dari 80% maka rekomendasi adalah *upgrade* kapasitas RAM menjadi 4 GB. Konfigurasi langsung 4 GB ke virtual *server*. Status 2.
4. Apabila kapasitas RAM adalah 4 GB dan peak prediksi kurang dari 80% dan lebih dari 10% maka rekomendasi adalah akan tetap. Konfigurasi tetap 4 GB ke virtual *server*. Status 2.
5. Apabila kapasitas RAM adalah 4 GB dan prediksi lebih dari 80% maka rekomendasi akan mengeluarkan Peringatan Penggunaan sumber daya melampaui batas. Konfigurasi tetap 4 GB ke virtual *server*. Status 2.



Gambar 6. Diagram Alir Rekomendasi

2.1.6 Diagram Alir Rekonfigurasi



Gambar 7. Diagram Alir Rekonfigurasi

Berdasarkan Gambar 7 diagram alir rekonfigurasi dijelaskan sebagai berikut.

1. Proses rekonfigurasi dilakukan dengan mengecek hasil rekomendasi terlebih dahulu.
2. Jika status rekomendasi 0, maka ubah konfigurasi jumlah memori.
3. Jika status rekomendasi 1, maka ubah konfigurasi jumlah memori.
4. Jika status rekomendasi 2, maka ubah konfigurasi jumlah memori.
5. Setelah direkonfigurasi, alur sistem akan kembali seperti semula.

2.2 Implementasi Sistem

Gambar 8. Implementasi Sistem

Berdasarkan Gambar 8 implementasi sistem ini dijelaskan sebagai berikut.

1. Proses pertama kali dilakukan untuk membangun sistem *monitoring* adalah terlebih dahulu menyiapkan satu buah PC.
2. Install OS dan konfigurasi untuk *hypervisor*, yang memiliki tugas yang penting untuk proses prediksi, rekomendasi dan rekonfigurasi memori.
3. Install OS dan konfigurasi untuk virtual *server*, sebagai dasar untuk dilakukan proses *monitoring*.
4. Lakukan pengecekan terhadap *hypervisor* dan virtual *server* apakah sudah sudah ter-install dengan baik atau masih ada yang kurang. Jika masih ada kesalahan, maka lakukan kembali tahap penginstalan.

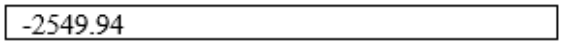
2.3 K-means Regresi

K-means regresi digunakan untuk melakukan proses prediksi pada memori server. Berdasarkan penggunaan memori, k-means sedemikian rupa mengatasi masalah dengan jumlah data banyak dengan tujuan agar data yang dihasilkan lebih simple dan lebih mudah dikelola oleh regresi linier. *Clustering* k-means digunakan untuk mencari nilai centroid/rata-rata dari hasil *monitoring*, dimana k-means menjadi nilai masukan untuk regresi linier. Hasil Centroid ditampilkan pada tabel 1.

Tabel 1 K-means Centroid

Waktu (detik)	*Used Memory* (KB)
467.5225	474308.1
211.9911	1269895
344.0933	201502.5
216.1951	999127.5
203.984	731732.9

Setelah mendapatkan hasil centroid, maka regresi linier akan melakukan proses prediksi untuk memori server kedepannya. Regresi linier menghasilkan sebuah koefisien yang ditampilkan pada gambar 9. Jika koefisien bernilai negatif maka grafik yang ditunjukkan akan menurun, sebaliknya jika koefisien bernilai positif maka grafik yang ditunjukkan akan menaik.



Gambar 9 Koefisien dari regresi

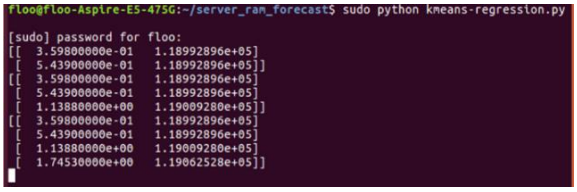
3. PENGUJIAN DAN ANALISIS

3.1 Pengujian *Monitoring*

Tujuan dari pengujian ini adalah untuk melihat hasil penggunaan memori dari virtual *server*. Hal pertama yang dilakukan adalah menjalankan *server* dengan perintah "*python monitor.py &*" yang kemudian akan dilakukan *monitoring* oleh *hypervisor*. Proses *server* telah berjalan ditampilkan pada Gambar 10.

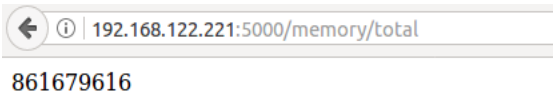
Gambar 10 *Server* telah berjalan

Setelah *server* berjalan, *hypervisor* kemudian melakukan *monitoring* di terminal dengan mengeksekusi script dengan perintah "*sudo python kmeans-regression.py"* untuk proses *monitoring*, prediksi, rekomendasi dan rekonfigurasi. *Monitoring* di *hypervisor* dapat dilihat pada Gambar 11.



Gambar 11 Proses *Monitoring* di *hypervisor*

Pada Gambar 11 diatas, proses *monitoring* sudah dilakukan di *hypervisor*. Kolom yang pertama sebelah kiri adalah waktu eksekusi secara *realtime* dalam detik. Penggunaan waktu *realtime* sebagai objek *monitoring* adalah karena penelitian dilakukan secara *realtime*/pada saat ini dan agar *hypervisor* mengetahui kebiasan atau kecenderungan dari memori *server*. Penjelasan kolom pertama sama dengan $3.598x10^{-1}$, berarti dimulai dari detik ke 0,3598, sama halnya dengan kolom yang dibawahnya. Kolom yang kedua sebelah kanan adalah pemakaian *used memory* dari virtual *server*. Penggunaan *used memory* sebagai objek *monitoring* adalah untuk melihat berapa banyak pemakaian memori pada saat ini. Dengan menggunakan *used memory* maka, *hypervisor* akan dapat memprediksikan pemakaian yang dibutuhkan untuk kedepannya. Kolom kedua sama dengan $1.18992896x10^{5}$, berarti pemakaian *used* memorinya sama dengan 118992,896 KB, sama halnya dengan kolom yang dibawahnya.

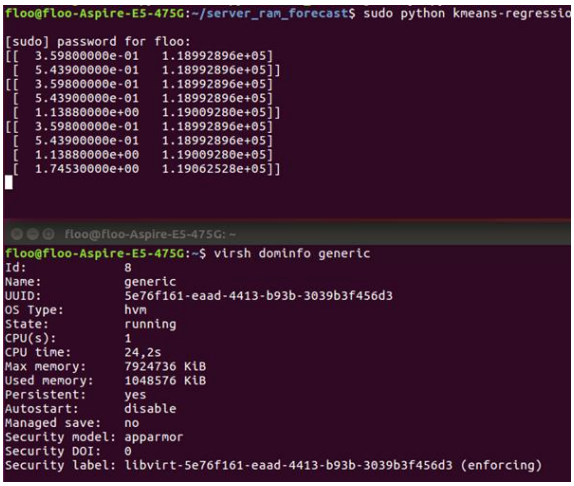


Gambar 12. *Monitoring* total di web *browser*

Pada Gambar 12 diatas, menjelaskan bahwa proses *monitoring* virtual *server* bisa melalui browser dengan menggunakan IP dan PORT dari virtual *server*. Gambar diatas adalah jumlah memori awal dari virtual *server* dengan menggunakan total memori sebagai kata kunci untuk menampilkan total memori dari virtual *server*. Jumlah memori diatas adalah 861679616 bytes, berarti total memori saat ini sama dengan 1 GB.

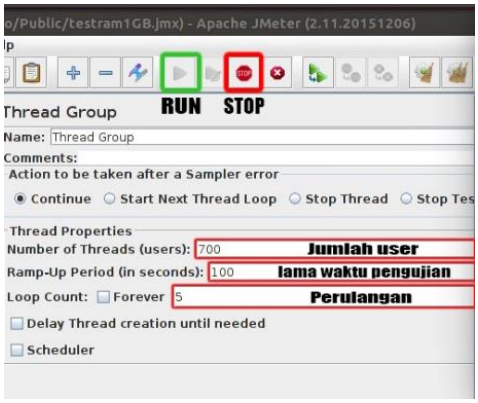
3.2 Pengujian Prediksi, Rekomendasi dan Rekonfigurasi

Dalam tahap ini yaitu proses *monitoring* dengan menjalankan script *kmeans-regression.py*. *Monitoring* akan dimulai dari waktu *realtime* dan *used memory*. *Hypervisor* juga bisa mendapatkan informasi dari virtual *server* dengan mengetikkan "*virsh dominfo generic"*. Proses *monitoring* dan informasi virtual server dapat dilihat pada Gambar 13.



Gambar 13. Proses *monitoring* di *hypervisor*

Untuk mendapatkan hasil *upgrade* dan waktu rekonfigurasi, tools *apache jmeter* sudah bisa di start. *Tools apache jmeter* seperti pada Gambar 14 dan 15.



Gambar 14. *Tools apache jmeter thread group*

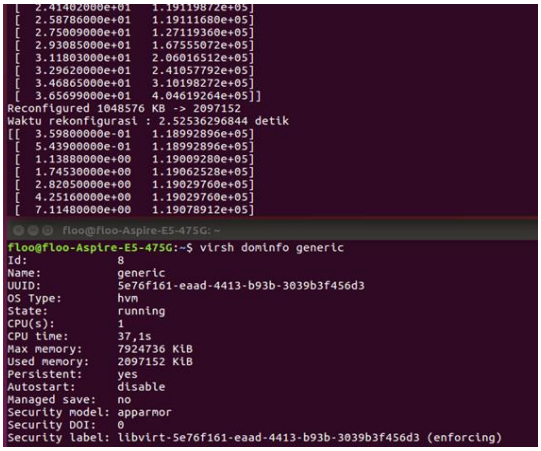
Gambar 15. Tools *apache meter* http *Request*

Dengan *tools apache jmeter*, sistem berhasil melakukan *upgrade* pada detik ke 36.5699 dan *used memory* menjadi 404019.264 KB yang awalnya total memori 1GB menjadi 2GB dan *hypervisor* juga berhasil melakukan rekonfigurasi ke virtual *server* menjadi 2GB.

Server_status sekarang bernilai 1 untuk 2 GB. Hasil upgrade, waktu rekonfigurasi dan analisis ditampilkan pada Tabel 2, Gambar 16 dan Tabel 3.

Tabel 2. Hasil prediksi dan rekomendasi skenario satu

Server_status	Memori (Kapasitas)
1	2097152 KB



Gambar 16. Hasil prediksi dan waktu rekonfigurasi 2GB

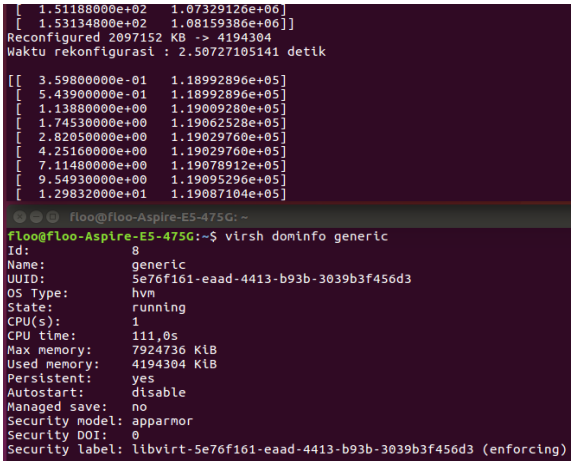
Tabel 3. Analisis *upgrade* dari 1 GB ke 2 GB

Proses *Upgrade* ke 2 GB		
Waktu real dan *used memory*	Waktu prediksi dan *used memory*	Waktu prediksi dan ram persen
36.5699 detik dan 404619.264 KB	96.5699 detik dan 851388.4 KB	96.5699 detik dan **81.19473%**

Pada gambar dibawah, detik ke 153.1348, *used memory* menjadi 1081593.856 KB dan *hypervisor* kembali melakukan *upgrade* dari 2 GB ke 4 GB dan *hypervisor* juga berhasil melakukan rekonfigurasi ke virtual *server* menjadi 4 GB. Server_status sekarang bernilai 2 untuk 4 GB. Hasil *upgrade* juga ditampilkan dalam "*virsh dominfo generic", used Memory* sudah di *upgrade* menjadi 4 GB.. Hasil *upgrade*, waktu rekonfigurasi dan analisis ditampilkan dalam Tabel 4, Gambar 17 dan Tabel 5.

Tabel 4. Hasil prediksi dan rekomendasi skenario satu

Server_status	Memori (Kapasitas)
2	4194304 KB



Gambar 17. Hasil prediksi dan waktu rekonfigurasi ke 4 GB

Tabel 5 Analisis *Upgrade* dari 2 GB ke 4 GB

Proses *Upgrade* ke 4 GB		
Waktu real dan *used memory*	Waktu prediksi dan *used memory*	Waktu prediksi dan ram persen
153.1348 detik dan 1081593.856 KB	213.1348 detik dan 1681352 KB	213.1348 detik dan **80.17311%**

Hasil *upgrade* juga bisa di cek dari browser dengan menggunakan IP dan PORT. Untuk menampilkan jumlah memori dari hasil *upgrade*, menggunakan perintah total. Jumlah memori pada skenario satu saat ini menjadi 4 GB.

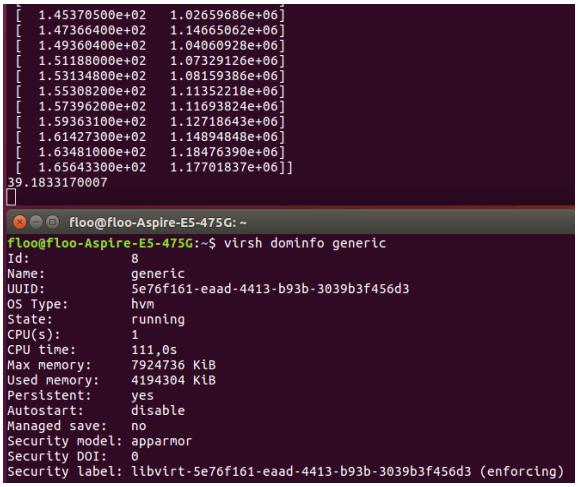
Gambar 18. Total memori virtual *server*

- **Skenario dua**

Pada skenario dua, kondisi memori sekarang adalah maksimum. Untuk melihat hasil *downgrade* dari memori, proses *request* ke *server* akan dihentikan. Proses *monitoring* saat ini turun karena tidak adanya *request*. Konfigurasi maksimum memori pada virtual *server* ditampilkan pada Tabel 6.

Tabel 6. Kondisi maksimum memori pada skenario dua

Server_status	Memori (Kapasitas)
2	4194304 KB

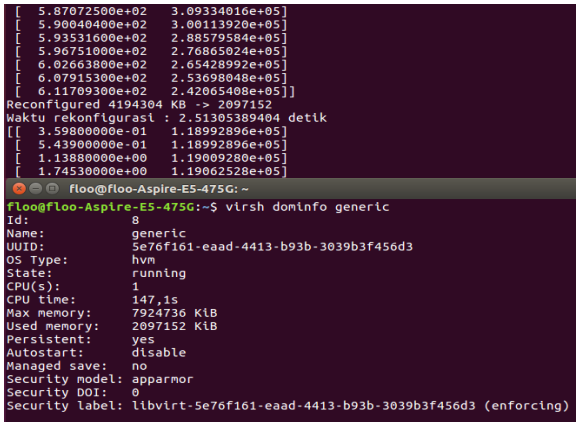


Gambar 19. Memori maksimum virtual *server*

Pada gambar dibawah, *Hypervisor* berhasil melakukan *downgrade* pada detik ke 611.7093 dan *used memory* saat ini adalah 242065.408 KB. Hasil *downgrade* dari 4 GB ke 2 GB. Server_status sekarang bernilai 1 untuk 2 GB. Hasil *downgrade*, waktu eksekusi dan analisis ditampilkan pada Tabel 7, Gambar 20 dan Tabel 8.

Tabel 7. Hasil Downgrade pada Skenario Dua

Server_status	Memori (Kapasitas)
1	2097152 KB



Gambar 20. Memori *downgrade* ke 2 GB

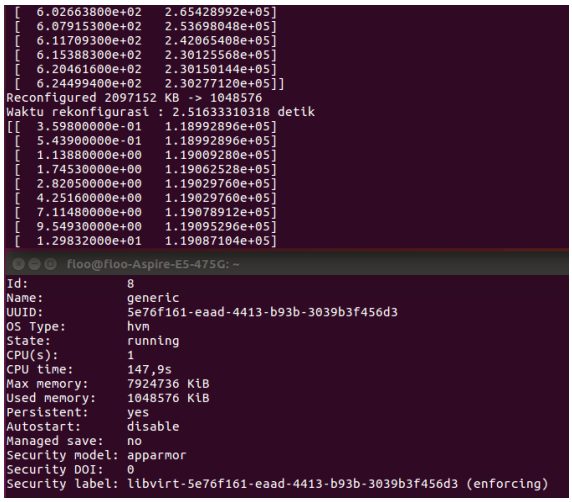
Tabel 8. Analisis *downgrade* dari 4 GB ke 2 GB

Proses Downgrade ke 2GB		
Waktu real dan *used memory*	Waktu prediksi dan *used memory*	Waktu prediksi dan ram persen
611.7093 detik dan 242065.408 KB	671.7093 detik dan 382828.8 KB	671.7093 detik dan **9.127349%**

Hypervisor kembali melakukan *downgrade* pada detik ke 624.4994 dan *used memory* saat ini adalah 230277.12 KB. Hasil *downgrade* dari 2GB ke 1GB. Server_status sekarang bernilai 0 untuk 1GB. Hasil *downgrade*, waktu eksekusi dan analisis ditampilkan pada Tabel 9, Gambar 21, Tabel 10.

Tabel 9. Hasil *downgrade* ke 1 GB pada skenario dua

Server_status	**Memori (Kapasitas)**
0	1048576 KB



Gambar 21 Memori d*owngrade* ke 1GB

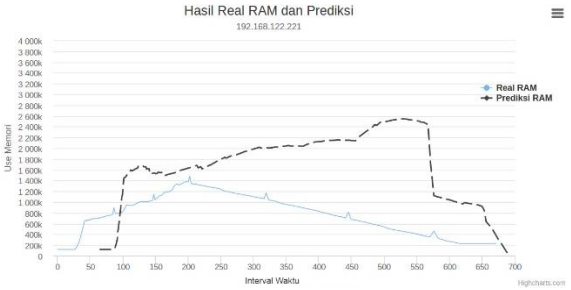
Tabel 10 Analisis *downgrade* dari 2 GB ke 1 GB

Proses *Downgrade* ke 1GB		
Waktu real dan *used memory*	Waktu prediksi dan *used memory*	Waktu prediksi dan ram persen
624.4994 detik dan 230277.12 KB	684.4994 detik dan 137128.3 KB	684.4994 detik dan **6.538787%**

Setelah semua skenario selesai dilakukan, dicek kembali di virtual *server* apakah memori padad virtual *server* berubah apa tidak. Untuk melihat jumlah memori, dengan mengetikkan perintah *"free"*. Hasil memori dari skenario yang telah dilakukan dapat dilihat pada Gambar 22.

Gambar 22 Total memori virtual s*erver*

Hasil dari proses prediksi akan ditampilkan dalam sebuah grafik Hasil *Real* RAM dan Prediksi RAM. Garis X adalah interval waktu dan grafik Y adalah *used memory*. Garis berwarna biru adalah *real ram* dan garis yang berwarna hitam adalah hasil prediksi. Pada gambar grafik penggunaan memori dan prediksi dibawah ini, hasil dari prediksi memiliki suatu perbedaan. *Real ram* pada detik ke 202.4875 menunjukkan grafik menurun dengan jumlah *server* sudah mencapai 4 GB. *Real ram* menurun dikarenakan proses *request* ke *server* telah berhenti, sehingga penggunaan memori server akan menurun. Sedangkan prediksi ram pada detik ke 262.4875 menunjukkan grafik naik. Prediksi naik dikarenakan oleh data pada saat monitoring memiliki data yang banyak, sehingga penggunaan k-means regresi perlu beberapa waktu untuk proses pengelompokan untuk mendapatkan prediksi sampai menunjukkan grafik menurun. Pada detik ke 536.0597, grafik dari prediksi sudah menurun seiring menurunnya proses *monitoring*. Grafik penggunaan memori dan prediksi seperti pada Gambar 23.



Gambar 23 Grafik Penggunaan Memori dan Prediksi

4. KESIMPULAN

Berdasarkan hasil pengujian dan analisis dari implementasi sistem ini, dapat ditarik kesimpulan dari penelitian ini, yaitu sebagai berikut.

1. Penulis telah mengimplementasikan sebuah sistem implementasi alokasi memori dinamis pada sistem komputer berbasis *clustering* k-means yang dibangun dalam satu computer fisik dengan cara virtualisasi menggunakan KVM *Hypervisor* dan membangun *server* dengan KVM/QEMU.
2. Proses alokasi memori dinamis menggunakan metode k-means regresi. K-means regresi dilakukan dengan cara mengelompokan data berdasarkan hasil dari proses *monitoring* yang menghasilkan sebuah centroid. Nilai centroid akan menjadi masukan untuk regresi linier yang kemudian akan dilakukan sebuah prediksi, rekomendasi dan rekonfigurasi untuk server. Regresi linier menghasilkan sebuah koefisien dengan nilai akhir -2549.94.
3. Alokasi memori dilakukan secara otomatis tanpa peran administrator. Sistem pada awalnya melakukan *monitoring* memori pada virtual *server* dengan keluaran waktu (detik) dan *used memory* (KB), kemudian masuk ke proses prediksi. Proses prediksi adalah proses dari hasil *monitoring* memori. Proses dari hasil prediksi menghasilkan nilai dalam bentuk persen(%). Prediksi yang melampaui batas yang ditentukan akan di *upgrade* sesuai dengan hasil rekomendasi dan prediksi yang kurang dari batas yang ditentukan akan di *downgrade* sesuai dengan hasil rekomendasi. Setelah dilakukan rekomendasi maka akan langsung di rekonfigurasi. Rekonfigurasi dilakukan untuk merubah jumlah memori pada virtual *server* sesuai dengan hasil dari rekomendasi. Waktu eksekusi rekonfigurasi ke virtual *server* membutuhkan waktu 2.5 detik. Pada grafik hasil prediksi menunjukkan bahwa semakin banyak data yang diterima oleh k-means regresi maka grafik dari prediksi akan menunjukkan grafik yang berbeda jauh dari hasil *real* ram server.

DAFTAR PUSTAKA

Awaris, Muhammad Abdullah. (2016). Memory Management: Challenges and Techniques for Traditional Memory Allocation Algorithms in Relation with Today's Real Time Needs. Diambil kembali dari ijmse: http://www.ijmse.org/Volume7/Issue3/paper3.pdf

Baktir, A.C., Kulahoglu, Y.C., Erbay, O., Metin, B. (2013). Server Virtualization in Information and Communication Technology Infrastructure in Turkey. IEEE

Kaciak, Rajok, Gufron. (2013). Pengertian Virtualisasi. Diambil kembali dari Dosen.gufron: http://dosen.gufron.com/artikel/pengertian-virtualisasi/8

Ketu, Shwet., Agarwal, Sonali. (2015). *Performance Enhancement of Distributed K-Means Clustering for Big Data Analytics Through Inmemory Computation* (hal. 318-324). IEEE.

MacKay, David. (2003). Chapter 20. An Example Inference Task: Clustering. *Information Theory, Inference and Learning Algorithms. Cambridge University Press.*

Niswar, M., Sabri, A. A., Warni, E., Musa, N. M. (2013). Memory Sharing Management on Virtual Private Server. IEEE

Oracle. (2004). How To Effectively Size Hardware for Your Portal Implementation. An Oracle White Paper.

Sattar, Rida. (2016). What is Apache Jmeter. Diambil kembali dari Toolsqa: http://toolsqa.com/jmeter/what-is-apache-jmeter/