ISSN : 2407 - 3911

MODEL PREDIKSI *SIMPLE MOVING AVERAGE* PADA *AUTO-SCALING CLOUD COMPUTING*

Novian Anggis Suwastika[1)], Praditya Wahyu W[2)], Tri Broto Harsono[3)]
Fakultas Informatika
Universitas Telkom Bandung
Jl Telekomunikasi Terusan Buah Batu, Dayeuhkolot Bandung 40257
Email : anggis@telkomuniversity.ac.id[1)], praditya wahyu@gmail.com[2)], tri.brotoharsono@gmail.com[3)]

Abstrak

Simple Moving Average (SMA) yang merupakan salah satu metode pada model sistem prediksi yang berbasis time series dengan karakteristik komputasinya yang sederhana dibandingkan dengan metode yang lain. Analisis model prediksi SMA ini akan diujikan pada kondisi stabil dan fluktuatif untuk melihat performansi model dengan parameter uji *Mean Time Between Failure (MTBF), Mean Time To Repair (MTTR), Availability Operational (AO), Down Time* dan *Up Time*. Hasil implementasi model nilai AO di atas 96%.

Kata kunci :

Cloud-computing, auto-scaling, sistem prediksi, simpel moving average

Abstract

Simple Moving Average (SMA), is one method for prediction system model time series based with a simple computational characteristics compared with other methods. Analysis of SMA prediction models will be tested in a stable condition and fluctuating to see the performance of the model with test parameters Mean Time Between Failure (MTBF), Mean Time To Repair (MTTR), Operational Availability (AO), Down Time and Up Time. Results of the implementation of the model AO value above 96%.

Keywords :

Cloud-computing, auto-scaling, prediction system, simple moving average.

I. PENDAHULUAN

Layanan *cloud* computing yang bersifat *on-demand service* dan *rapid-elasticy* (Buyya dkk, 2011) banyak dipilih sebagai suatu layanan yang memberikan kemudahan di jaringan internet saat ini terutama dalam penyimpanan data dan komputasi. *Auto-scaling* menjadi salah satu cara agar *cloud* yang dibangun memiliki karakteristik *on-demand service* dan *rapid-elasticity* karena *auto-scaling* dapat mengoptimalkan layanan *cloud* berdasarkan prediksi ataupun kondisi tertentu yang ditentukan, hal tersebut menjadikan *auto-scaling* bersifat *scalable* dan *self adaptive*. Salah satu metode *auto-scaling* adalah dengan sistem prediksi, metode ini menggunakan suatu algoritma dalam memprediksi penggunaan *resource* di sisi server sehingga sistem *cloud computing* yang dibangun dapat memperkirakan berapa jumlah *resource* yang dibutuhkan oleh layanan yang dijalankannya.

Terdapat banyak metode sistem prediksi yang dapat diimplementasikan pada *auto-scaling* untuk menjaga efisiensi *resource* server bagi kebutuhan terhadap layanan yang dijalankan, salah satu jenisnya adalah berbasis *time series* atau keturutan waktu, dimana metode ini menggunakan sejumlah data dimasa lalu sebagai acuan untuk melakukan prediksi selanjutnya. Sistem prediksi *time series* dibagi lagi menjadi beberapa jenis, salah satunya adalah *simple moving average*. SMA memiliki karakteristik komputasi yang sederhana dibandingkan dengan metode yang lain, karena metode ini menggunakan nilai rataan sejumlah data masa lalu dengan menentukan jumlah periode tertentu untuk menentukan nilai prediksi yang akan datang (Yafee dan McGee, 2002).

Struktur artikel ini adalah sebagai berikut, pembahasan mengenai komponen dasar penelitian terdapat pada bab dua, hasil penelitian mengaju pada parameter uji dan sekanrio pengujian serta analisis hasil implementasi dijelaskan pada bab tiga dan kesimpulan pada bab empat

II. KAJIAN LITERATUR

II.1 *Cloud Computing*

Cloud computing dapat didefinisikan sebagai sebuah cara baru komputasi yang menerapkan konsep *scalability* dan *virtualization* untuk penggunaan *resource* komputasi pada jaringan internet. Dengan menggunakan layanan *cloud computing*, pengguna yang menggunakan berbagai macam *device* seperti PC/Laptop, smartphone dan PDA dapat mengaksses program, media penyimpanan dan application-development platforms di atas jaringan internet, melalui *service* yang diberikan penyedia layanan *cloud computing*. Keuntungan dari teknologi *cloud computing* diantaranya adalah:

1. *Cost saving* bagi pengguna, karena pengguna tidak perlu membeli atau menyediakan server untuk komputasi.
2. *High availability*, *resource* selalu tersedia dimanapun dan kapanpun pengguna membutuhkan.
3. *Easy scalability*, pengguna dapat mengatur jumlah *resource* yang dibutuhkan sesuai kebutuhan pengguna.

Pada teknologi *cloud computing* layanan yang diberikan dapat dibedakan menjadi beberapa layer arsitektur service.Layer atas adalah layanan *cloud computing* yang disebut dengan *Software as a Service* (SaaS), layanan ini memungkinkan pengguna untuk melakukan *running* aplikasi secara remot terhadap *cloud.* Layer layanan berikutnya adalah *Infrastructure as a Service*(Iaas) yang menyediakan penggunaan computing *resource* sebagai *service cloud*, termasuk virtualisasi komputer yang menjamin *processing power* dan *bandwidth* untuk *storage* atau media penyimpanan dan internet akses.

Layer *Platform as a Service* (PaaS) memiliki layanan yang hampir serupa dengan IaaS, perbedaannya adalah PaaS menyediakan layanan *operating system* dan beberapa *service* yang diperlukan untuk menjalankan beberapa aplikasi. Sederhananya PaaS adalah IaaS dengan fitur untuk *custom software stack* yang dapat digunakan untuk menjalankan suatu aplikasi. *Layer* layanan yang terakhir adalah *data Storage as a Service* (dSaaS) yang memungkinkan penyimpanan data bagi pengguna termasuk kebutuhan *bandwidth* bagi media penyimpanannya.

Untuk kemampuan akses layanan, teknologi *cloud computing* dibedakan menjadi beberapa jenis yaitu *publi ccloud*, *private cloud*, serta *hybrid cloud*. Pada *public cloud* atau eksternal *cloud*, *resource* untuk *computing* secara dinamis dikelola di atas jaringan internet melalui *web application* atau *web services* dari suatu off-site pihak ketiga yang memberikan layanan. *Public cloud* dijalankan oleh pihak ketiga dan aplikasi dari pengguna yang berbeda-beda yang saling bercampur bersama pada *cloud server*, sistem penyimpanan maupun jaringan.

Private cloud atau internal cloud merupakan layanan cloud computing yang dijalankan pada *private networks*, *private cloud* ini dibangun untuk exclusive client yang memungkinkan untuk melakukan full-control terhadap data, aplikasi, *security* maupun *quality of service*. Private cloud ini dapat dibangun dan dikelola oleh pemilik perusahaan, organisasi atau kelompok IT, atau bahkan cloud provider itu sendiri.

Hybrid cloud mengkombinasikan model *multiple public* dan *private cloud*, yang dapat mengakomodir bagaimana distribusi aplikasi diantara sebuah *public cloud* dengan *private cloud* (Furt dan Escalante, 2010).

II.2 *Auto-scaling*

Auto-scaling merupakan suatu teknik yang digunakan untuk mengelola *resource* komputasi secara otomatis, sehingga administrator tidak perlu melakukan penambahan ataupun pengurangan *resource* secara *manual* ketika menjalankan suatu layanan *cloud* (Ferraris dkk, 2012). Penerapan *auto-scaling* ini akan meningkatkan efektivitas dan efisiensi *resource* komputing *cloud* maupun mengurangi beban administrator *cloud.*

Sistem *auto-scaling* dapat dijalankan dengan berbagai *rule* sehingga proses *scaling*-nya dapat dijalankan, ada dua jenis *rule* yang digunakan dalam *auto-scaling* ini yaitu *predictably*(prediksi) dan

dynamically (Buyya dkk, 2011). Metode prediksi menggunakan aturan dengan melakukan prediksi terhadap layanan yang dijalankan, dengan melakukan prediksi akan diketahui kapan kira-kira terjadi lonjakan trafik ataupun penurunan trafik yang kemudian akan dilakukan *scaling* yang disesuaikan dengan trafik yang dibutuhkan, misalnya suatu ketika trafik naik secara drastis maka CPU *utilization* dapat ditingkatkan menjadi 50% dari normal, *Memory usage* ditingkatkan, atau bahkan penambahan *resource* dari server lainnya dan seterusnya. Sistem prediksi sangat efektif jika *predictivesystem* yang digunakan tepat, namun sebaliknya, sistem prediksi ini dapat memperburuk penggunaan *resource* jika *predictivesystem* yang digunakan tidak tepat.

Dynamically melakukan proses *scaling* secara dinamis tidak tergantung dari trafik yang berjalan, metode ini akan melakukan *scaling* untuk layanan-layanan tertentu yang telah diatur oleh administrator, misalnya ketika sistem menjalankan layanan dari Amazon EC2 maka CPU *utilization* ditingkatkan beberapa persen, memory ditingkatkan menjadi beberapa gigabytes, port-port tertentu dibuka, dan sebagainya. Metode ini akan baik jika memang layanan tersebut ketika dijalankan membutuhkan *resource* yang sesuai, namun jika layanan tersebut dijalankan sedangkan pemakaian layanan sangat sedikit maka hal tersebut dapat mengurangi efisiensi penggunaan *resource* dan menambah biaya dari *resource* yang berjalan.

II.3 Metode SimpleMoving Average (SMA)

SMA merupakan salah satu jenis metode prediksi berdasarkan*time series* atau keturutan waktu kuantitatif dalam teori peramalan. Metode SMA menggunakan nilai pada masa lalu untuk digunakan sebagai acuan dalam melakukan prediksi pada masa depan. Secara umum tujuan dari jenis peramalan *time series* adalah menemukan pola dalam deret historis dari suatu data dan mengeskploitasinya untuk dijadikan pola masa depan.

Data *time series* seringkali mengandung ketidakteraturan yang akan menyebabkan prediksi yang beragam. Untuk menghilangkan efek yang tidak diinginkan dari ketidak-teraturan ini, metode SMA mengambil beberapa nilai yang sedang diamati, memberikan rataan, dan menggunakannya untuk memprediksi nilai untuk periode waktu yang akan datang (Yaffe dan McGee, 2000). Semakin banyak jumlah pengamatan yang dilakukan, pengaruh metode *moving average* akan semakin baik. Meningkatkan jumlah observasi akan menghasilkan nilai peramalan yang lebih baik karena ia cenderung meminimalkan efek-efek pergerakan yang tidak biasa yang muncul pada data (Linuxinfo, 2006).

Metode SMA dapat dirumuskan sebagai berikut (Botrano dkk, 2012):

$$A_t = \frac{D_t + D_{t-1} + D_{t-2} + \cdots + D_{t-N+1}}{N} \ldots\ldots\ldots (1)$$

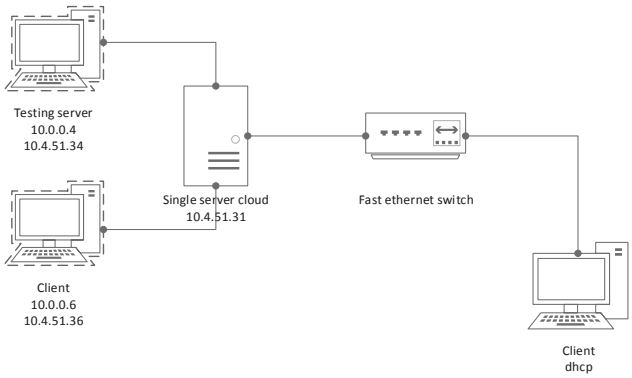
Dengan:

N adalah total jumlah periode rataan.

A_t adalah prediksi pada periode $t+1$.

II.4 Perancangan Sistem Cloud Computing

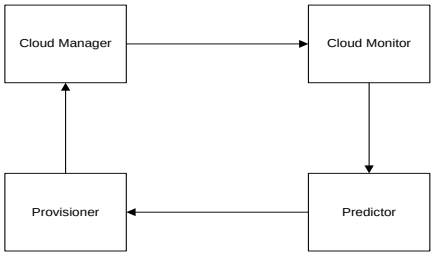
Pada penelitian ini *cloud computing* yang dibangun dengan menggunakan framework OpenStack (Kevin, 2012).



Gambar 1. Topologi Jaringan *Cloud Computing*

Arsitektur topologi cloud yang akan digunakan ditunjukkan pada Gambar 1. Satu *server* digunakan untuk membangun sistem *cloud, Server testing* untuk membangun *web server* dan *video streaming* server untuk melakukan pengujian dan satu buah komputer client. Ketiga komponen terhubung dengan roiuter untuk menghubungkan dengan jaringan internet. Untuk spesifikasi masing-masing komponen ditunjukkan pada Tabel 1. Untuk model rancangan model *auto-scaling* yang dibangun untuk implementasi SMA ditunjukkan pada Gambar 2.

Predictor akan melakukan prediksi terhadap *resource* yang harus digunakan dalam melayani suatu permintaan dari pengguna. Pada sistem akan didefinisikan *treshold* yang digunakan sebagai batasan untuk proses scaling, *treshold* atas didefinisikan untuk melakukan *scale up* dan *treshold* bawah didefinisikan untuk melakukan *scaledown*. Ketika suatu data atau informasi yang dikirim melewati batas *treshold* maka *triger* akan aktif untuk melakukan proses *scaling*, script yang berisikan prediksi untuk pengalokasian *resource* akan diaktifkan yang dikirimkan dari provisioner kepada *cloud* manager sehingga *cloud* manager akan membuat alokasi *resource* sesuai dengan apa yang diminta oleh *provisioner*, akhirnya *cloud* manager mengalokasikan *resource* untuk layanan yang dijalankan.



Gambar 2. Model *auto-scaling* untuk SMA

Tabel 1. Spesifikasi Komponen Infrastruktur

Komponen	Server Cloud	Client
Processor	Intel i7	Core2Duo
Memory	6GB	3GB
Hardisk	1TB	50 GB
Aplikasi	OpenStack	Menyesuaikan

Layanan yang berjalan di *cloud* akan selalu dimonitor oleh *cloud* monitor sehingga dapat diketahui apakah perlu melakukan *scale up* atau *scale down*. *Treshold* atas pada penelitian ini dapat didefinisikan sebesar 80% dari total memory, sedangkan untuk *treshold* bawah pada sistem yang dibangun ini didefinisikan dengan 10% total memory.

Setelah pembangunan sistem, disusun skenario pengujian untuk meneliti pengaruh dari auto-scaling dengan *predictive system* pada *cloud computing* yang dibangun. Performansi auto-scaling akan diketahui setelah melakukan pengujian terhadap sistem dan seberapa tepat prediksi dari *simple moving average* yang diimplementasikan untuk menjalankan *auto-scaling*.

Pada penelitian ini skenario pengujian dilakukan pada permintaan layanan yang fluktuatif dan bersifat relatif stabil. Pengujian secara fluktuatif adalah yang penggunaan layanan pada *cloud* selalu berubah-ubah tidak tentu naik-turunnya, seperti *video streaming* atau *social media*, sedangkan pengujian yang relatif stabil dapat dilakukan dengan penggunaan aplikasi yang cenderung stabil penggunaanl ayanannya pada *cloud*, seperti penyimpanan dan pengelolaan data.

Skenario pengujian untuk penelitian ini adalah sebagai berikut:

1. Pengujian yang bersifat fluktuatif : *webserver* yang dibuat seolah-olah penggunaannya tidak tentu setiap waktunya. Durasi pengujian dilakukan dalam selang waktu 1-2 jam
2. Pengujian yang bersifat stabil : *web server* yang aktifitasnya cenderung sama setiap waktunya.

Dari pengujian sistem diharapkan dapat diperoleh data yang cukup untuk penelitian selanjutnya. Data yang akan diambil dari pengujian yang dilakukana dalah *uptime* dan *downtime server*, *Mean Time To Repair* (MTTR) yang menggambarkan waktu rata-rata dari server untuk dapat melakukan *recovery* layanan setelah mengalami *failure* atau *down*, *Mean Time Between Failure* (MTBF) yang menjelaskan jumlah rata-rata waktu dari server failure yang pertama dengan server failure yang selanjutnya, dan yang terakhir adalah *Availability Operational* (AO) yang dapat menjelaskan tentang lama operasional yang dapat dapat dijalankan dari server. Serta seberapa tepat penggunaan resource yang dialokasikan dengan metode simple moving average pada auto-scaling tersebut.

III. HASIL DAN ANALISIS

III.1 Hasil Pengujian Bersifat Fluktuatif

Pada pengujian yang bersifat fluktuatif, dilakukan dengan dua skenario yang berbeda, pertama dengan menggunakan jumlah periode 3 pada *auto-scaling* yang diimplementasikan dan yang kedua dengan jumlah periode 6 pada *auto-scaling*nya. Pengujian dilakukan selama kurang lebih 2 jam, dalam waktu tersebut dilihat seberapa baik *auto-scaling* yang dijalankan dengan memperhatikan jumlah *down*, lama waktu *down time*, akurasi prediksi

penggunaan *memory*, serta jumlah *uptime* saat menjalankan pengujian. *State* awal dari pengujian ini adalah dengan menggunakan *instance* webserver yang memiliki jumlah *memory* atau RAM sebesar 1024 MB atau 1 GB dengan jumlah core CPU yaitu 1 buah. Setelah dilakukan pengujian selama waktu yang telah ditentukan masing-masing dengan menggunakan kedua periode yang dibandignkan tersebut diperoleh hasil :

Tabel 2.Perbandingan Hasil Pengujian Fluktuatif

No	Subject	Jml periode 6	Jml periode 3
1	jml down time/failure	2	2
2	failure time	220 detik	270 detik
3	available time	7999detik	7999detik

III.2 Hasil Pengujian Bersifat Stabil

Pada pengujian ini dibandingkan penggunaan memory dengan resource usage yang seolah-olah stabil di sisi server. Pengujian yang dilakukan untuk skenario ini tidak terlalu lama seperti yang dilakukan pada pengujian fluktuatif, karena dengan resource usage yang selalu stabil atau tidak ada naikan atau turunan secara signifikan maka, sebagian pengujian akan mewakili keseluruhan karakteristik dari sistem yang dijalankan. State awal dari pengujian ini sama seperti yang dilakukan pada pengujian fluktuatif, yaitu dengan 1024MB RAM dan 1 buah core CPU pada sisi server.

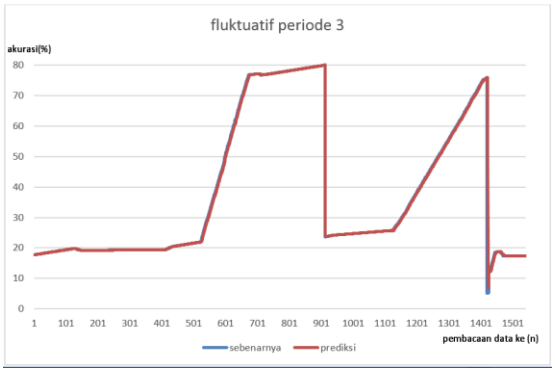
Dari pengujian yang dilakukan diperoleh hasil seperti pada tabel 3.

Tabel 3.Hasil Pengujian Stabil

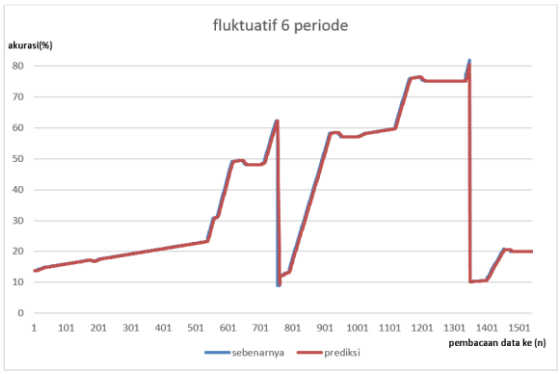
no	Subject	jml periode 3	jml periode 6
1	jml down time/failure	0	0
2	failure time	0 detik	0 detik
3	available time	1740 detik	1740 detik

III.3 Analisis Pengujian Bersifat Fluktuatif

Dari pengujian dengan membuat seolah-olah penggunaan memory berjalan secara fluktuatif, didapatkan nilai prediksi dan nilai sebenarnya dari penggunaan memory pada server yang diujikan, berdasarkan nilai yang didapatkan tersebut dapat dibuat suatu grafik perbandingan keduanya seperti dapat dilihat pada gambar 3 dan 4.



Gambar 3. Hasil pengujian fluktuatif 3 Periode



Gambar 4. Hasil pengujian fluktuatif 6 Periode

Jika diperhatikan secara detail didapatkan hasil bahwa auto-scaling dengan prediksi *simple moving average* terlihat selalu mengikuti hasil sebenarnya dari penggunaan *resource* memory pada webserver. Ketika penggunaan memory mengalami kenaikan prediksi yang didapatkan selalu berada di bawah nilai sebenarnya walaupun tidak terpaut jauh, namun ketika penggunaan *resource* memory sebenarnya mengalami penurunan, maka prediksi yang didapatkan selalu di atas nilai sebenarnya. Dari rumus SMA [rumus (1)] dapat dianalisis bahwa memang prediksi yang dilakukan dengan menggunakan *simple moving average* akan selalu "tertinggal" jika mengalami kenaikan atau penurunan aktifitas, karena metode ini menggunakan beberapa nilai berdasarkan jumlah periode yang digunakan, sebagai bahan untuk mendapatkan prediksi selanjutnya. Misalkan ketika aktifitas naik, maka SMA juga akan menggunakan nilai-nilai sebelum proses kenaikan tersebut terjadi oleh sebab itulah mengapa prediksi yang dihasilkan oleh metode ini akan tertinggal ketika mengalami kenaikan aktifitas, begitu juga ketika aktifitas turun, metode prediksi

SMA juga akan menggunakan nilai sebelum proses penurunan itu terjadi, hal itu juga menjadi dasar kenapa prediksi SMA ketika terdapat penurunan aktifitas akan selalu di atas nilai sebenarnya.

Berdasarkan pengujian tersebut dapat diperoleh nilai MTTR, MTBF serta *availability operational* :

1. MTBF = (available time)/(number of failure)
   SMA dengan 6 periode : 3999,5 detik
   SMA dengan 3 periode : 3999,5 detik

2. MTTR = (failure time)/(number of failure)
   SMA dengan 6 periode : 110 detik
   SMA dengan 3 periode : 135 detik

3. *Availability operational* =(MTBF)/(MTBF+MTTR)
   SMA dengan 6 periode : 0,9732 atau 97,32 %
   SMA dengan 3 periode : 0,9673 atau 96,73 %

Dari perhitungan tersebut diperoleh hasil bahwa SMA dengan 6 periode menghasilkan nilai *availability operational* yang lebih besar dibandingkan SMA dengan 3 periode pada pengujian yang telah dilakukan. Dengan kata lain didapatkan informasi bahwa ketersediaan layanan pada server *cloud* dengan *auto-scaling* yang mengimplementasikan SMA menggunakan jumlah periode lebih besar akan memberikan layanan yang lebih baik dibandingkan dengan SMA yang menggunakan jumlah periode lebih kecil. Namun hal tersebut tidak sepenuhnya dapat diterima sebagai kesimpulan akhir karena pada implementasi auto-scaling ini terjadi proses *resizing instance* ketika proses *scaling* dijalankan. Ketika *resizing instance* dilakukan maka layanan yang dijalankan pada server yang sedang melakukan *resizing* tersebut untuk sementara tidak dapat diakses, dan akan dapat diakses kembali ketika proses *resizing* telah selesai.

Proses *resizing* itulah yang sebenarnya menentukan hasil dari *availability operational*, karena proses *resizing* oleh openstack ini tidak sama satu dengan yang lain, namun secara umum proses ini terjadi selama 1-3 menit. Dengan kata lain ketika proses *resizing* terjadi dalam waktu yang lebih lama maka secara otomatis akan mengurangi waktu ketersediaan layanannya, begitu juga sebaliknya. Pada penelitian ini penggunaan kedua periode menghasilkan jumlah proses *scaling* yang sama yaitu dua kali dengan kata lain resizing yang dilakukan juga sebanyak dua kali namun hasil *availability operational* yang didapatkan tidak sama atau ada sedikit selisih antara keduanya. Dari informasi tersebut akan lebih tepat jika tidak menyatakan bahwa jumlah periode SMA yang digunakan pada proses auto-scaling tidak memiliki pengaruh langsung terhadap ketersediaan layanan atau *availability operational.*

Akurasi prediksi yang dilakukan dengan metode *simple moving average* dihitung dengan membandingkan nilai antara nilai sebenarnya dengan nilai prediksi yang dilakukan, kemudian melakukan rata-rata berdasarkan jumlah data yang ada, dari 1540 data yang didapatkan berdasarkan pengujian yang dijalankan, dilakukanlah perhitungan untuk menentukan berapa persen akurasi dari prediksi yang dihasilkan oleh metode *simple moving average* pada kasus ini. Metode MAPE[6] digunakan untuk mengetahui hasil akurasi, yang selanjutnya akan diperoleh nilai akurasi rata-rata dari 1540 data yang ada sehingga diperoleh informasi :

Nilai rata-rata akurasi SMA dengan 6 periode : 97,59019684 %

Nilai rata-rata akurasi SMA dengan 3 periode : 96,81408525 %

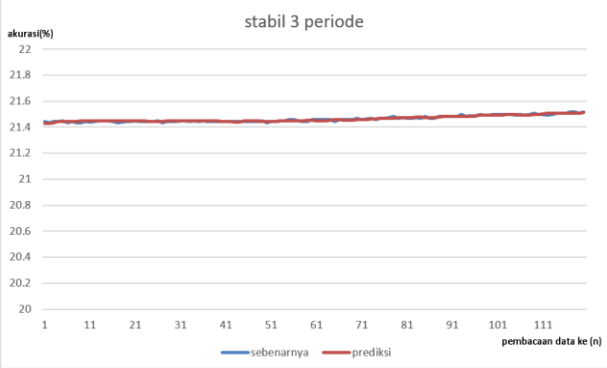
Berdasarkan informasi yang didapatkan akurasi SMA dengan 6 periode lebih tinggi dibandingkan SMA dengan periode 3. Berdasarkan hasil tersebut teori mengenai semakin besar jumlah periode akan mempengaruhi akurasi prediksi menjadi lebih besar begitu pula sebaliknya semakin kecil periode yang digunakan pada metode SMA maka hasil akurasinya akan semakin berkurang atau semakin kecil[16] terpenuhi. Walaupun hasil yang didapatkan tidak terlalu berbeda jauh karena memang periode pembanding yang digunakan hanya dua kali lipatnya, namun hasil ini sedikit menjelaskan bahwa memang penggunaan jumlah periode akan mempengaruhi seberapa akurat prediksi yang dihasilkan dengan *auto-scaling* yang dijalankan.

III.4 Analisis pengujian bersifat stabil

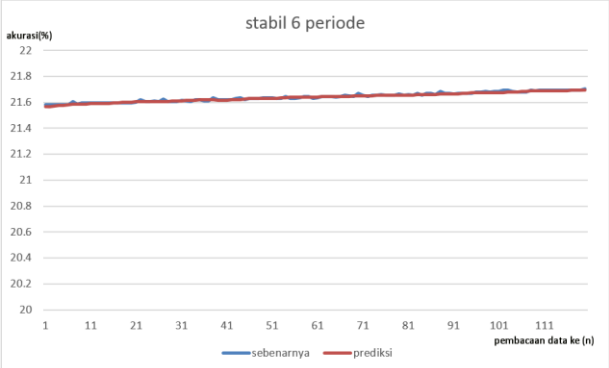
Hasil analisis pengujian bersifat stabil dapat dilihat pada gambar 5 dan 6.

Jika dilihat secara sekilas maka hasil prediksi dengan nilai sebenarnya yang ada pada server tidak memperlihatkan perbedaan yang begitu mencolok. Pada pengujian seolah-olah penggunaan *memory* stabil maka nilai MTTR, MTBF maupun *availability operational* dapat dilihat hasilnya tanpa harus melakukan perhitungan seperti yang dilakukan pada

pengujian fluktuatif, dengan tidak adanya *failure* atau *down* pada pengujian ini maka secara otomatis nilai dari *availability operational* adalah 100%. Selanjutnya melakukan penilaian terhadap keakuratan prediksi antara SMA dengan jumlah periode yang berbeda.



Gambar 5. Hasil Pengujian Stabil 3 Periode



Gambar 6. Hasil Pengujian Stabil 6 Periode

Dari hasil perhitungan berdasarkan 120 data yang diperoleh dengan cara perhitungan yang sama seperti pada pengujian fluktuatif, didapatkan informasi sebagai berikut:

- Akurasi prediksi SMA dengan 3 periode : 99.9794086 %
- Akurasi prediksi SMA dengan 6 periode : 99.97478051 %

Berdasarkan informasi yang diperoleh di atas, akurasi yang dihasilkan oleh *simple moving average* dengan jumlah periode 3 sedikit lebih besar daripada akurasi yang didapatkan dengan jumlah periode 6. Perbedaan yang sangat sedikit tersebut hampir dapat dikatakan sama atau tidak terlihat. Hal tersebut diperoleh karena memang aktifitas penggunaan *resource* yang terjadi di dalam server tidak begitu signifikan perubahannya dan cenderung statis. Karena itulah secara tidak langsung dapat dikatakan bahwa dengan periode berapapun akan menghasilkan hasil yang tidak berbeda jauh karena kestabilan penggunaan *resource* tadi. Hasil tersebut diperoleh karena metode auto-scaling dengan SMA ini menggunakan beberapa nilai masa lalu sebagai acuan sedangkan nilai antara masa lalu dengan yang dihasilkan selanjutnya tidak banyak terdapat perbedaan, jadi berapa besar periode untuk pengujian relatif stabil ini tidak terlalu berpengaruh pada akurasi yang dihasilkan.

Pada kasus aktifitas yang relatif stabil maupun fluktuatif, *threshold* atas atau batas atas dari prediksi selain dapat ditentukan sendiri oleh penguji juga dapat disesuaikan dengan keadaan sistem saat itu, sehingga *threshold* tersebut lebih ideal bagi sistem yang dijalankan. Perhitungan penggunaan *threshold* yang ideal bagi sistem dapat dilakukan dengan menggunakan fungsi pengurangan dari alokasi *memory* keseluruhan pada server dikurangi dengan jumlah *memory* yang terpakai saat itu, jika di dalam linux dapat ditulis dengan " free -m | awk '/Mem/{print (($2-$3)/$2)*100}' ". Dengan perhitungan tersebut akan didapatkan batas dari *threshold* atas yang ideal digunakan untuk melakukan prediksi penggunaan *memory* pada server yang diuji. Hal tersebut karena ketika *threshold* yang didefinisikan tidak lebih dari sisa *memory* yang telah digunakan oleh sistem maka secara tidak langsung ketika prediksi telah mencapai batas yang ditentukan sistem secara otomatis akan melakukan peningkatan jumlah *memory*, dan hal tersebut tidak akan mengganggu kestabilan sistem yang diuji.

Pada penelitian ini fungsi yang digunakan untuk melakukan pengambilan informasi resource *memory* dari sistem adalah dengan menggunakan command "free", command ini sering digunakan pada Linux atau Unix-like untuk mengetahui informasi used atau free *memory* yang ada. Jika diperhatikan baris pertama dari command ini jika dijalankan akan menunjukkan informasi seputar penggunaan *memory* yang termasuk alokasi terhadap buffer dan caches, buffer *memory* biasanya didefiniskan sebagai bagian dari *memory* yang di set sebagai penyimpanan temporary untuk data yang dikirimkan atau diterima dari external device.Pada baris kedua berisi informasi seputar buffer cache yang digunakan saat itu, cache ini digunakan oleh program untuk melakukan akses file sehingga dapat mempercepat saat akses pada data yang informasinya terdapat pada cahce tersebut.

Sementara pada baris ketiga terdapat informasi penggunaan swap oleh sistem jika ada. Namun pada prakteknya free akan menginformasikan pengukuran resource *memory* sedikit lebih kecil dari kenyataannya, hal tersebut karena terdapat penggunaan dari kernel yang tidak dapat di swap out, penggunaan oleh kernel akan selalu tertinggal di main *memory* ketika komputer dalam keadaan beroperasi, dengan demikian ketika ada yang menempati atau mengambil sebagian tempat di main *memory* maka *memory* yang digunakan tidak dapat dibebaskan. Ada juga baigan dari *memory* yang dicadangkan untuk keperluan yang lain sesuai dengan arsitektur sistem tertentu (Roy dkk, 2011). Dari pemaparan tersebut penggunaan command free sebagai dasar untuk mengambil informasi resource yang digunakan pada server tidaklah salah karena memang informasi yang ditunjukkan tidaklah 100% akurat karena ada penggunaan lain pada *memory* yang tidak dapat dilihat informasinya.

IV. KESIMPULAN

Akurasi prediksi metode SMA yang diimplementasikan pada sistem *cloud computing* dengan pengujian fluktuatif dipengaruhi oleh besarnya periode yang digunakan, semakin besar akan semakin akurat. Sementara nilai availability operational tidak dipengaruhi secara langsung oleh jumlah periode SMA yang digunakan namun lebih dipengaruhi oleh lama proses *scaling* yang dijalankan. Hasil dari paremeter uji AO berada pada treshold atas dan bawah dengan hasil di atas 96% begitu juga akurasi prediksi penggunaan *resource*.

REFERENSI

Furt, Borko, dan Escalante, Armando, 2010, Handbook of Cloud Computing, London, Springer.

Ferraris, Filippo Lorenzo, Franceschelli, Davide , dan Gioiosa, Mario Pio, 2012, Evaluating the Auto Scaling Performance of Flexiscale and Amazon EC2 Clouds, Politecnico di Milano

Jackson, Kevin, 2012,Openstack Cloud Computing Cookbook, Birmingham, PACKT Publishing

Linuxinfo, The free command.(2006)[Online] tersedia di : http://www.linfo.org/free.html

Roy, Nilabja, Dubey, Abhishek, dan Gokhale, Aniruddha, 2011, Efficient Autoscaling in the Cloud using Predictive Models forWorkload Forecasting, Vanderbilt University.

Buyya, Rajkummer, Broberg, James, dan Goscinski, Andrzej, 2011, Cloud Computing Priciples and Paradigms, United State, Wiley.

Yaffee, Robert A. , dan McGee, Monnie ,2000, Introduction to Time Series Analysis and Forecasting: With Applications of SAS and SPSS, New York, Academic Press

Botrano, Tania Lorido, Alonso, Jose Miguel, dan Lozano, Jose A.,2012,Auto-scaling Techniques for Elastic Applications in Cloud Environments,University of the Basque Country.

University of Guelph. 2013. Forecasting. http://www.uoguelph.ca/~dsparlin/forecast.htm# CAUSAL FORECASTING METHODS