SELF ADAPTIVE SOFTWARE DENGAN TEKNIK *CASE-BASED REASONING* UNTUK MENINGKATKAN PERFORMA *APPLICATION SERVER*

[1]Muhamad Nur, [2]Domo Pranowo Kuswandono
[1,2]Program Studi/Jurusan Teknik Informatika, STMIK Bani Saleh Bekasi

ABSTRACT

Performance of application server is an important attribute that must be maintained quality and stability, because application server operated in dynamic environment with fluctuating user loads and resource levels, its potentially caused unpredictable errors. In order application server performance still optimal, it is need maintains configuration application parameters. But, do that process need deep observation each environment condition changed and of course the cost and time. The approach that can be used is self-adaptive (autonomic computing). Self-adaptive can configure application parameters automatically. Best combination application parameters configuration can be stored on a repository and used as reference when decide new decision for configuration of application server parameters when similar condition is occurred. Cased-based reasoning can do the process. Best approach will be used is self-adaptive with case-based reasoning. This research implements the approach on an experiment application that deployed on glassfish application server. The results show that self-adaptive with case-based reasoning can improve application server performance with significant improvement.

Keywords: *Self adaptive Software, Self-configuring, CBR, Framework, Application server*

ABSTRAK

Performa pada sebuah application server adalah atribut penting yang harus dijaga kualitas dan stabilitasnya, karena application server beroperasi pada lingkungan kerja yang dinamis dan mempunyai fluktuasi tinggi dalam hal user load dan level resource sehingga sangat berpotensi terjadi error. Agar performa application server tetap optimal, maka perlu dilakukan perubahan konfigurasi parameter application server. Tapi untuk melakukan hal itu membutuhkan observasi setiap ada perubahan kondisi lingkungan kerja dan tentu saja biaya dan waktu. Salah satu pendekatan yang bisa digunakan adalah menggunakan self-adaptive atau biasa disebut dengan istilah autonomic computing. Self-adaptive dapat melakukan proses tuning performa application server secara otomatis. Kombinasi terbaik konfigurasi parameter application server dapat disimpan pada repository untuk dijadikan acuan ketika mengambil keputusan perubahan konfigurasi saat mengalami perubahan kondisi yang mirip dengan kondisi yang pernah terjadi. Teknik case-based reasoning (CBR) dapat melakukan proses tersebut. Karena itu pendekatan terbaik yang bisa digunakan adalah self-adaptive berbasis case-based reasoning. Penelitian ini mengimplementasikan pendekatan diatas pada sebuah aplikasi eksperimen yang dideploy pada glassfish application server. Hasil yang didapat menunjukan bahwa self-adaptive dengan teknik penentu keputusan case-based reasoning mampu meningkatkan performa application server dengan prosentase cukup signifikan.

Kata kunci: *Self adaptive Software, Self-configuring, CBR, Framework, Application server*

1. PENDAHULUAN

1.1. Latar Belakang

Application server atau server aplikasi adalah perangkat lunak yang menyediakan lapisan servis standar untuk mendukung pengembangan, penerapan dan pendistribusian aplikasi yang berjalan pada sisi server. Contoh teknologi server aplikasi adalah J2EE dan .NET yang bisa digunakan untuk membangun server aplikasi dan mampu memberikan berbagai kualitas layanan. Performa pada sebuah server aplikasi adalah atribut penting yang harus dijaga kualitas dan stabilitasnya, karena server aplikasi beroperasi pada lingkungan kerja yang dinamis dan mempunyai fluktuasi yang tinggi dalam hal user load dan level resource sehingga sangat berpotensi terjadi error yang sulit diprediksi (Liu, Gorton, & Trivedi, 2006).

Agar performa server aplikasi tetap pada level tinggi dan stabil, maka dibutuhkan pengelolaan yang intensif saat berjalan pada sebuah lingkungan kerja, misalnya maintenance atribut server sesuai dengan perubahan yang terjadi (Laddaga, 1999) (Kephart & Chess, 2003). Pengelolaan tersebut dilakukan oleh administrator yang terus memonitor performa server aplikasi saat sedang berjalan.

Setiap perubahan yang terjadi dan mempengaruhi performa server, maka administrator akan melakukan proses tuning performa server aplikasi tersebut misalnya dengan merubah ukuran thread pool, durasi time out transaksi dan jumlah maksimum koneksi yang terhubung secara bersamaan (Liu & Gorton, 2007). Tapi untuk menentukan pengaturan parameter atribut terbaik sangat sulit, dan harus dilakukan melalui observasi dan percobaan setiap terjadi perubahan lingkungan kerja serta membutuhkan biaya yang tidak sedikit (Jordan, Czajkowski, Kouklinski, & Skinner, 2004) (Gorton, Liu, & Brebner, 2003). Terutama pada perusahaan yang memiliki ketergantungan terhadap IT yang sangat tinggi seperti perbankan.

Salah satu solusi yang tepat adalah dengan menggunakan teknologi autonomic computing untuk mengotomatisasi manajemen performa server aplikasi ketika sedang berjalan. *Autonomic computing* atau *self adaptive* dapat melakukan secara mandiri proses tuning performa server aplikasi yang dilakukan oleh administrator. Untuk menentukan konfigurasi yang tepat saat proses tuning, parameter yang digunakan lebih dari satu dan kombinasi parameter tersebut menentukan performa server aplikasi. Bentuk kombinasi parameter tersebut dapat dijadikan kasus yang disimpan dan digunakan sebagai acuan untuk menentukan kombinasi parameter terbaik. Salah satu pendekatan yang dapat digunakan adalah *Case-Based Reasoning*.

Case-Based Reasoning adalah teknik pemecahan masalah dengan memanfaatkan kasus pengalaman yang pernah terjadi yang sesuai dan efisien dengan kasus yang sedang dihadapi (Kolodner, 1992). Pengambilan manfaat-manfaat yang baru dari kasus yang telah terjadi membuat Case-Based Reasoning lebih atraktif dan elegan dibandingkan dengan teknik pemecahan masalah yang lain (Khan, Awais, & Shamail, 2008). Pada penelitian ini akan dicoba mengimplementasikan self adaptive software dan Case-Based Reasoning dalam bentuk sebuah framework yang bisa digunakan untuk meningkatkan performa dan stabilitas server applikasi khususnya java application server.

1.2. Identifikasi Masalah

Permasalahan yang akan yang mendasari penelitian ini adalah sebagai berikut:

1. Performa application server kurang optimal dalam merespon request dari client dalam sebuah lingkungan kerja yang dinamis.
2. Kondisi lingkungan kerja server sering berubah sehingga tidak sesuai dengan konfigurasi parameter application server dan membutuhkan perubahan konfigurasi ulang.
3. Sulit menentukan konfigurasi parameter terbaik untuk menjaga performa application server tetap optimal karena memerlukan observasi khusus setiap terjadi perubahan lingkungan kerja dan untuk melakukannya membutuhkan biaya yang besar.
4. Penerapan perubahan konfigurasi parameter application server membutuhkan restart sehingga dapat menggangu proses bisnis sistem yang sedang berjalan.

1.3. Ruang Lingkup Masalah

Ruang lingkup masalah penelitian ini adalah bagaimana cara meningkatkan performa application server agar tetap stabil dalam sebuah kondisi lingkungan kerja yang sering berubah dalam merespon request dari client. Karena itu, masalah akan dibatasi hanya sebatas bagaimana membuat sebuah komponen dalam bentuk framework self-adaptive yang bisa digunakan untuk membantu application server melakukan self-configuring terhadap atribut-atribut yang diperlukan agar server tetap berjalan dalam performa tinggi dan stabil. Proses self-configuring yang akan digunakan diambil dari hasil perbandingan kasus yang telah lampau menggunakan algoritma similarity dalam system case-based reasoning.

1.4. Tujuan Penelitian

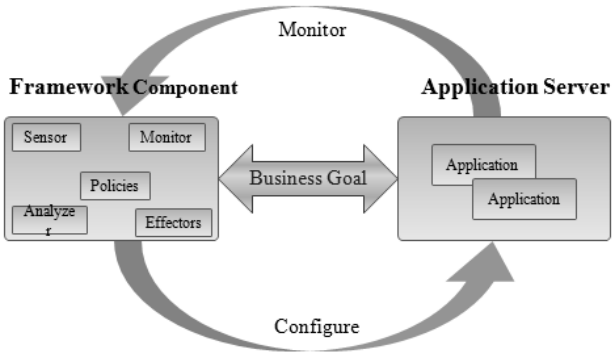
Tujuan penelitian ini adalah meningkatkan performa application server dengan mengembangkan komponen framework self-adaptive menggunakan metode case-based reasonin. Metode tersebut digunakan untuk menentukan keputusan ketika melakukan self-configuring terhadap terhadap property application server. Penelitian ini dapat dimanfaatkan sebagai salah satu alternaatif strategi manajemen sebuah server aplikasi yang membutuhkan stabilitas dan peningkatan performa saat sedang berjalan dalam sebuah lingkungan kerja. Hasil penelitian ini akan menyediakan salah satu pendekatan yang diwujudkan dalam bentuk framework yang dapat digunakan dalam menjaga kualitas performa sekaligus kontrol secara otonomi application server.

2. METODOLOGI PENELITIAN

Metode Penelitian yang digunakan pada penelitian ini adalah model eksperimen instrumentasi. Penelitian ini bertujuan untuk meningkatkan performa java application server dengan mengembangkan sebuah framework self-adaptive berbasis case-based reasoning. Jenis data yang digunakan pada penelitian ini adalah data primer yang diambil dari hasil observasi terhadap performa java application server yang bekerja pada sebuah lingkungan kerja yang dinamis. Penelitian ini diarahkan untuk membuktikan bahwa framework self-adaptive berbasis case-based reasoning yang dikembangkan dapat meningkatkan performa java application server. Tahapan yang digunakan pada penelitian ini adalah sebagai berikut:

2.1. Arsitektur Software

Sasaran utama dari dibuat dan digunakannya framework ini terhadap application server adalah untuk memisahkan implementasi perubahan konfigurasi termasuk semua adaptive behavior server dengan logika bisnis dari applikasi yang berjalan pada server. Implementasi adaptive ini menggunakan komponen framework yang dijalankan dalam sebuah adaptive engine.



Gambar 1. Arsitektur Software

Komponen pada engine tersebut berinteraksi dengan aplikasi dan membuat perubahan konfigurasi secara dinamis terhadap adaptive behavior server berdasarkan data yang diterima. Interaksi bisa dilakukan dalam bentuk memonitor jalannya server, menganalisa data yang dikumpulkan sensor dan eksekusi perubahan konfigurasi baik secara real time maupun secara antrian tegantung dari perubahan yang dibutuhkan. Misalnya mengatur koneksi yang sedang aktif dan mengatur jumlah thread yang sedang berjalan.

2.2. Model Performance

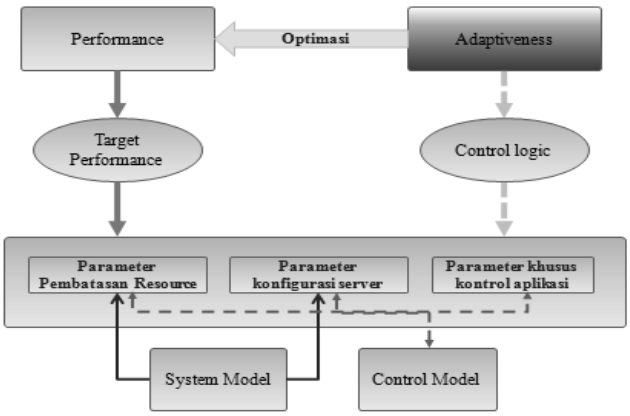
Untuk bisa menentukan nilai sebuah performa, maka performa tersebut harus bisa dipresentasikan dalam bentuk besaran quantitatif yang bias diukur. Ada tiga aspek yang harus dipertimbangkan ketika akan mempresentasikan performa sebuah application server, yaitu:

1. Konversi tujuan atau sasaran performa application server menjadi bilangan yang bisa diukur besarannya. Bilangan tersebut bisa menjadi target output yang akan dicapai pada tahap analisis.
2. Tentukan pengaruh behavior yang akan diadaptasi dengan performa server. Berapa besar pengaruh perubahan konfigurasi yang diadaptasi terhadap performa application server.
3. Kalibrasi faktor-faktor application server seperti parameter konfigurasi server dengan performance server yang sudah dimodelkan. Misalnya atur nilai thread-pool-size pada application server pada nilai yang telah ditentukan pada model dan tentukan performance maksimum yang bisa didapat.

Aspek-aspek yang telah disebutkan diatas bisa dijadikan bahan untuk membuat strategi pengembangan lebih lanjut. Pendekatannya adalah dengan membagi variabel dinamis dari application server menjadi beberapa group kategori yaitu parameter khusus kontrol applikasi, parameter konfigurasi server dan parameter pembatasan resource server.

2.3. Parameter Khusus Kontrol Aplikasi

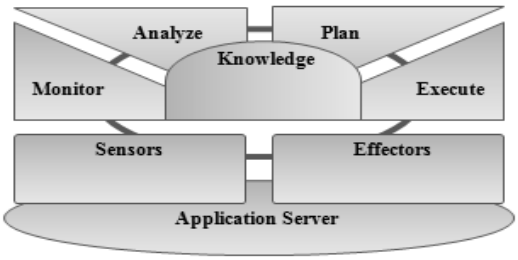
Parameter khusus kontrol aplikasi adalah parameter yang menunjukan behavior yang bisa diadaptasi. Misalnya pengadaptasian terhadap ukuran dokumen gambar yang direquest oleh client. Jika kecepatan koneksi jaringan cepat, berapapun ukuran dokumen gambar tidak menjadi masalah serius. Tapi jika kecepatan koneksi jaringan sangat lambat tentu akan membebani application server ketika memberikan response dimana data yang besar dipaksa untuk melewati bandwidth yang kecil. Adaptasi terhadap ukuran dokumen gambar pada jaringan dengan kecepatan lambat diperlukan agar response server tidak terhambat dan tetap dapat bekerja pada performa stabil. Kecepatan jaringan client tentu sangat bervariasi, hal ini berimplikasi terhadap settiingan pada server. Jika parameter yang diberikan besar, maka konfigurasi akan tidak sesuai jika client yang request mempunyai kecepatan network yang lambat. Tapi sebaliknya jika parameter yang diberikan kecil, maka settingan akan tidak sesuai jika client yang melakukan request mempunyai kecepatan yang cepat.



Gambar 2. Hubungan antara performance dan proses adaptasi

2.4. Pengembangan Siklus Kontrol

Setelah analisis aspek performance digambarkan dengan model, maka langkah selanjutnya adalah mendesain dan mengimplementasikan komponen framework sebagai siklus kontrol. Sebuah referensi framework standar tentang siklus kontrol pada autonomic computing pernah dikemukakan oleh Autonomic Computing IBM. Sebuah siklus kontrol terdiri dari monitoring yang berfungsi sebagai pemantau jalannya aplikasi, analyze yang berperan menganalisa data yang berhasil dikumpulkan, plan yang menangani rencana respon yang akan diberikan, dan execution yang akan mengambil tindakan adaptasi terhadap adaptive behavior. Setelah analisis aspek performance digambarkan dengan model, maka langkah selanjutnya adalah mendesain dan mengimplementasikan komponen framework sebagai siklus kontrol. Sebuah referensi framework standar tentang siklus kontrol pada autonomic computing pernah dikemukakan oleh Autonomic Computing IBM. Sebuah siklus kontrol terdiri dari monitoring yang berfungsi sebagai pemantau jalannya aplikasi, analyze yang berperan menganalisa data yang berhasil dikumpulkan, plan yang menangani rencana respon yang akan diberikan, dan execution yang akan mengambil tindakan adaptasi terhadap adaptive behavior.

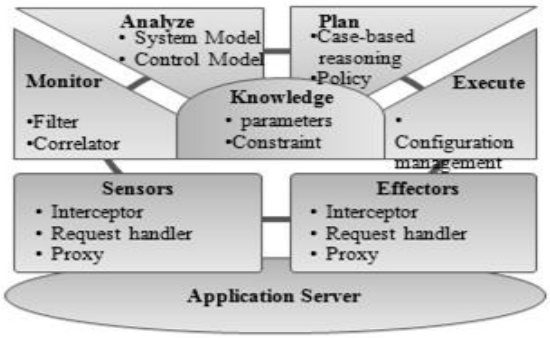


Gambar 3. Siklus kontrol dalam arsitektur standar autonomic computing

Siklus kontrol dikendalikan oleh informasi atau data yang terkait dengan performa server. Informasi performa server tersebut bisa berupa data-data settingan parameter konfigurasi server yang nilainya bisa dipantau atau dimonitor selama server dalam kondisi berjalan pada lingkungan kerja. Adapun komponen-komponen yang berada pada siklus tersebut antara lain:

1. Sensors
   Adalah komponen yang mendeteksi, menangkap, mengukur dan mengumpulkan data dengan metode sampling tanpa menganalisa terlebih dahulu. Misalnya CPU usage, kecepatan jaringan, penggunaan memory dan frekuensi request yang masuk pada server. Semua data yang ditangkap diteruskan oleh sensor sebagai input pada komponen selanjutnya.

2. Monitor
   Adalah komponen yang memantau keadaan system server secara detail. Dalam monitor, data-data dari sensor dikorelasikan dan difilter oleh fungsi-fungsi monitoring.
3. Analyzer
   Adalah komponen yang bertugas untuk memfasilitasi system dan control model untuk menganalisa situasi untuk memutuskan apakah sebuah perubahan perlu diambil untuk mengoptimasi perform server atau tidak.
4. Plan
   Adalah komponen yang akan membuat strategi bagai mana sebuah tindakan diambil. Pada penelitian ini, komponen plan akan menggunakan pendekatan case-based reasoning. Pada pendekatan ini solusi diambil dari solusi kasus-kasus yang pernah terjadi. Untuk lebih detail pembahasan tentang hal ini akan dibahas pada bagian eksperimen.
5. Execute
   Adalah komponen yang menyediakan mekanisme untuk melaksanakan eksekusi kontrol komponen, misalnya penyetelan perubahan parameter konfigurasi server.



Gambar 4 Siklus kontrol dalam self-adaptive software

6. Effector
   Adalah komponen yang merupakan implementasi dari execute. Dalam efector proses adaptive behavior server dieksekusi. Efector merupakan manifestasi kinerja adaptive software dalam siklus ini. Setelah effector maka siklus akan kembali pada sensor dan demikian seterusnya.

2.5. Penggunaan Case-based Reasoning

Untuk menentukan solusi yang akan digunakan pada bagian effector, data yang diterima dari bagian monitoring direpresentasikan dalam bentuk sebuah kasus. Selanjutnya solusi akan ditentukan dengan teknik case-based reasoning yaitu mencari kasus lama yang tersimpan pada case repository yang mempunyai kemiripan paling tinggi dengan kasus baru. Solusi kasus yang mempunyai similarity paling tinggi akan diambil sebagai solusi kasus yang baru.

Atribut-atribut yang akan digunakan pada case-based reasoning telah didefinisikan pada bagian siklus model. Atribut-atribut tersebut yang akan dijadikan kriteria dalam mencari similarity antara kasus baru dengan kasus lama yang tersimpan pada case repository. Atribut-atribut tersebut adalah sebagai berikut:

Tabel 1. Atribut penentu Performa server

NO	Atribut
1	Ukuran gambar
2	Kecepatan koneksi jaringan
3	CPU usage
4	Request rate

Masing-masing atribut mempunyai nilai atribut yang akan dijadikan ukuran dalam penentuan berapa nilai yang dimilikinya. Nilai masing-masing atribut dan ukurannya adalah sebagai berikut:

Tabel 2. Nilai dan ukuran atribut

NO	Atribut	Nilai Atribut	Ukuran
1	Ukuran gambar	Small	< 10 KB
		Small Medium	10 KB ~ 100 KB
		Medium	100 KB ~ 500 KB
		Medium Large	500 KB ~ 1 MB
		Large	> 1 MB
2	Kecepatan koneksi jaringan	Slow	< 1 Kbps
		Slow Medium	1 Kbps ~ 100 Kbps
		Medium	100 Kbps ~ 500 Kbps
		Medium Fast	500 Kbps ~ 2 Mbps
		Fast	> 2 Mbps
3	CPU usage	Low	< 20 %
		Low Medium	20 % ~ 40 %
		Medium	40 % ~ 60 %
		Medium High	60 % ~ 80 %
		High	> 80 %
4	Request rate	Low	< 10 rps
		Low Medium	10 rps ~ 20 rps
		Medium	20 rps ~ 30rps
		Medium High	30 rps ~ 40 rps
		High	> 40 rps

Untuk menentukan nilai kedekatan atau similarity antara kasus baru dengan kasus lama yang tersimpan dalam repository, maka nilai masing-masing atribut dibandingkan. Hasil perbandingan menunjukan nilai antara 0 sampai dengan 1. Nilai 0 artinya perbandingan tidak menunjukan kesamaan sama sekali atau mutlak berbeda dan nilai 1 artinya perbandingan menunjukan kesamaan yang sangat tinggi atau sangat mirip.
Berikut adalah proses penghitungan nilai similarity (Kusrini & Luthfi, 2009):

$$Similarity(p,q) = \frac{\sum_{i=1}^{n} f(pi,qi)wi}{wi}$$

Similirity dua kasus

Keterangan:
p = Kasus baru
q = Kasus lama yang tersimpan dalam database
n = Jumlah atribut yang ada dalam kasus
I = Atribut individu antara 1 sampai dengan n
f = Fungsi Similarity antara kasus baru p dengan kasus lama q
w = Bobot yang diberikan pada atribut ke i

Sesuai dengan rumus diatas maka untuk menghitung similarity kasus baru dengan kasus lama adalah sebagai berikut:

Similarity = ((B1 * 1) + (B2 * 0.75) + (B3 * 0.75) + (B4 * 0.5))/(1 + 0.75 + 0.75 + 0.5)
atau
Similarity = ((B1 * 1) + (B2 * 0.75) + (B3 * 0.75) + (B4 * 0.5))/ 3

Keterangan:
B1 = Bobot nilai atribut "Ukuran gambar" perbandingan kasus baru dengan kasus lama
B2 = Bobot nilai atribut "Kecepatan koneksi jaringan perbandingan kasus baru dengan kasus lama
B3 = Bobot nilai atribut "CPU Usage" perbandingan kasus baru dengan kasus lama
B4 = Bobot nilai atribut "Request rate" perbandingan kasus baru dengan kasus lama

2.6. Teknik Analisis

Teknik analisis yang digunakan pada penelitian ini adalah perbandingan antara hasil pengukuran yang didapatkan sebelum menggunakan framework dengan hasil pengukuran yang didapat setelah menggunakan framework. Perbedaan antara kedua pengukuran tersebut akan memberikan gambaran berapa nilai peningkatan performa application server setelah menggunakan framework yang dibuat. Semakin tinggi nilai yang dihasilkan maka akan semakin tinggi nilai peningkatan performa yang didapat.

3. HASIL DAN ANALISA

3.1. Hasil

Self-adaptive software diimplementasikan pada sebuah aplikasi image gallery, dimana aplikasi tersebut dideploy pada sebuah application server. Saat server menerima request dari client, kualitas network client tentu berbeda-beda. Hal ini mempengaruhi response time server yang diterima oleh cilent. Agar server tetap stabil dalam memberikan response time, maka self-adaptive berperan dengan merubah konfigurasi response request berdasarkan parameter yang sudah dijelaskan. Keputusan konfigurasi server sebagai efector berdasarkan dari sensor yang terus memonitor kondisi server yang sedang berjalan

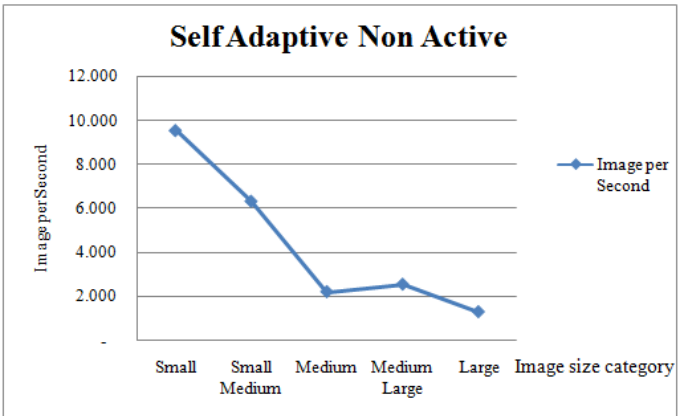
3.2. Analisa

Metode pengukuran yang digunakan pada penelitian ini adalah dengan mengimplementasikan aplikasi eksperimen pada sebuah personal computer (PC) sebagai server dan menggunakan 20 personal computer sebagai client yang akan melakukan request berupa file gambar dari server. Selanjutnya setiap aktivitas request client akan dicatat oleh server dalam sebuah log. Pengukuran pertama dilakukan saat kondisi aplikasi tidak mengaktifkan self-adaptive. Client melakukan request gambar yang sudah disediakan diserver dengan berbagai ukuran. Berikut adalah hasil pengukuran yang tercatat pada tabel log:

Tabel 3. Hasil pengukuran self-adaptive tidak aktif

Category	Milisecond per Image	Image per Second	Self-Adaptive
Small	104.636	9.557	Non-Active
Small Medium	158.000	6.329	Non-Active
Medium	454.000	2.203	Non-Active
Medium Large	393.000	2.545	Non-Active
Large	776.000	1.289	Non-Active

Data pada tabel diatas adalah data yang dicatat pade log yang menunjukan rata-rata waktu yang dibutuhkan oleh application server untuk merespon setiap request yang dilakukan oleh client. Terlihat bahwa masing-masing ukuran memerlukan respon time berbeda tergantung dari besarnya ukuran gambar yang diminta. Response time yang paling cepat adalah gambar yang mempunyai kategori ukuran paling kecil yaitu small, waktu yang dibutuhkan hanya 104.636 milisecond per image atau 6.329 image per second. Sedangkan yang memerlukan response time paling besar adalah gambar dengan kategori ukuran paling besar yaitu large, waktu yang dibutuhkan adalah 776 milisecond per image atau 1.289 image per second. Jika dirata-rata, maka response time yang diberikan adalah 377.127 second per image atau 4.384 image per second.



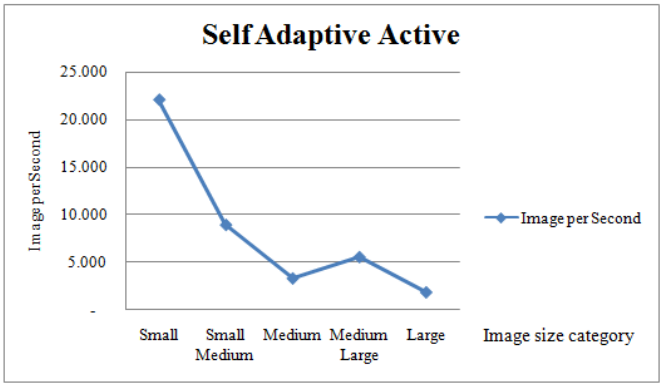
Gambar 5. Grafik hasil pengukuran self-adaptive tidak aktif

Pengukuran kedua dilakukan saat kondisi aplikasi mengaktifkan self-adaptive. Client melakukan request gambar yang sudah disediakan diserver dengan berbagai ukuran. Berikut adalah hasil pengukuran yang tercatat pada tabel log: Pengukuran kedua dilakukan saat kondisi aplikasi mengaktifkan self-adaptive. Client melakukan request gambar yang sudah disediakan diserver dengan berbagai ukuran. Berikut adalah hasil pengukuran yang tercatat pada tabel log:

Tabel 4. Hasil pengukuran self-adaptive aktif

Category	Milisecond per Image	Image per Second	Self-Adaptive
Small	45.273	22.088	Active
Small Medium	112.133	8.918	Active
Medium	304.455	3.285	Active
Medium Large	180.778	5.532	Active
Large	548.750	1.822	Active

Pada pengukuran self-adaptive diaktifkan, response time application server terasa lebih cepat dari sebelumnya. Hal ini bisa diamati dari hasil yang dicatat pada tabel log. Untuk kategori ukuran gambar paling kecil yaitu small, response time yang dibutuhkan hanya 45.273 milisecond per image atau 22.088 image per second. Sedangkan untuk kategori ukuran gambar paling besar yaitu large, response time server rata-rata menunjukan hanya 548.750 milisecond per image atau 1.822 image per second. Jika dirata-rata, maka response time yang diberikan adalah 238.278 milisecond per image atau 8.329 image per second.



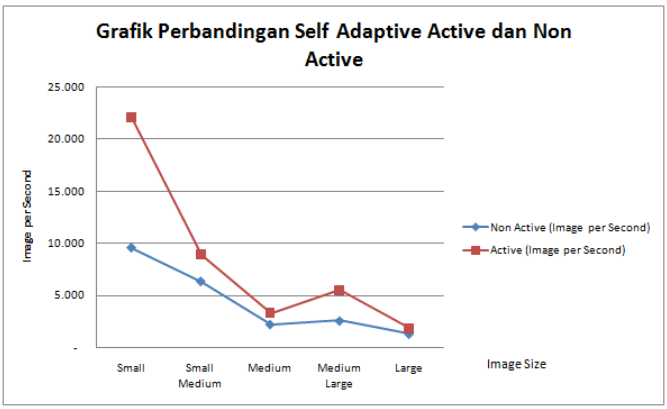
Gambar 6. Grafik hasil pengukuran self-adaptive aktif

Dari data hasil pengukuran kondisi self-adaptive tidak aktif dan kondisi self-adaptive aktif yang telah dilakukan, dapat dibandingkan dengan mengamati tabel berikut:

Tabel 5. Perbandingan hasil pengukuran self-adaptive aktif dan tidak aktif

Category	Non Active (mspi)	Non Active (ips)	Active (mspi)	Active (ips)	Improvement (%)
Small	104.636	9.557	45.273	22.088	131.12
Small Medium	158.000	6.329	112.133	8.918	40.90
Medium	454.000	2.203	304.455	3.285	49.12
Medium Large	393.000	2.545	180.778	5.532	117.39
Large	776.000	1.289	548.750	1.822	41.41

mspi → milisecond per image ips → image per second



Gambar 7. Grafik pengukuran self-adaptive aktif dan tidak aktif

Terlihat bahwa terjadi peningkatan di semua kategori ukuran gambar dengan nilai peningkatan yang bervariasi. Nilai peningkatan ini bisa diukur dengan prosentase yang tertera pada kolom improveent pada tabel. Peningkatan yang paling besar terjadi pada kategori ukuran gambar small yaitu sebesar 113. 12 %. Sedangkan peningkatan paling kecil terjadi pada kategori ukuran gambar smallmedium yaitu sebesar 40.90 %. Jika dirata-rata, maka prosesntase peningkatan yang dihasilkan adalah 75.99 %.

4. KESIMPULAN DAN SARAN

4.1. Kesimpulan

Berdasarkan hasil penelitian yang dilakukan dari tahap awal hingga eksperimen saat self-adaptive aktif dan tidak aktif dapat disimpulkan bahwa performa application server menjadi lebih optimal dan meningkat. Hal ini dibuktikan dengan analisis hasil pengujian eksperimen yang menunjukan response time mengalami peningkatan dari rata-rata 377.127 milisecond per image atau 4.384 image per second saat self-adaptive tidak diaktifkan menjadi 238.278 milisecond per image atau 8.329 image per second saat self-adaptive diaktifkan. Prosentase peningkatan bervariasi dan didapat rata-ratanya adalah 75.99 %.

Berdasarkan hasil diatas, maka secara umum didapat beberapa kesimpulan antara lain:

1. Pengaturan parameter konfigurasi aplication server dapat dilakukan secara autonomic dengan menerapkan pendekatan self-adaptive, sehingga performa application server bisa tetap dalam kondisi optimal.
2. Perubahan parameter konfigurasi application server dapat dilakukan tanpa harus melakukan proses restart, sehingga tidak menggangu proses bisnis dari system yang sedang berjalan.
3. Case-based reasoning dapat digunakan untuk menentukan pencarian solusi kasus baru pada saat self-adaptive akan melakukan proses effector dalam menentukan konfigurasi yang tepat berdasarkan data yang diterima dari berbagai sensor sebagai bentuk adaptasi dari perubahan kondisi lingkungan kerja.
4. Algoritma nearest neighbor dalam system cbr dapat mempermudah pencarian solusi yang sesuai dengan kasus yang sedang dihadapi dan membantu self-adaptive dalam menentukan konfigurasi yang tepat dalam melakukan proses adaptasi.

4.2. Saran

Self-adaptive software dengan case-based reasoning yang dikembangkan pada penelitian ini baru mencakup ruang lingkup java application server. Oleh karena itu diharapkan penelitian lebih lanjut akan dapat mengembangkan framework pada ruang lingkup yang lebih luas. Penerapan perubahan konfigurasi application server pada penelitian ini juga hanya menitik beratkan pada konfigurasi beberapa parameter application dan diharapkan dimasa yang akan datang dapat dilakukan penelitian yang mampu melakukan perubahan pada parameter yang lebih banyak. Berdasarkan dari apa yang telah dicapai pada penelitian ini, maka beberapa saran dibawah ini hendaknya bisa menjadi masukan pada penelitian selanjutnya yaitu antara lain:

1. Penerapan self-adaptive software tidak terbatas hanya pada java application server, tapi bisa juga digunakan pada application server lain atau software lain dengan karakter dan tugas fungsi yang berbeda.
2. Implementasi properties self-adaptive software yang diterapkan pada penelitian ini adalah self-configuring, pada penelitian selanjutnya bisa diterapkan self-properties lain.
3. Pendekatan yang digunakan pada penelitian ini menggunakan case-based reasoning, untuk penelitian selanjutnya bisa dicoba dengan pendekatan lain yang mungkin bisa menghasilkan hasil yang berbeda.

Daftar Pustaka

[1] Dobson, S., Denazis, S., Fernandez, A., & Gaiti, D. (2006). A Survey of Autonomic Communications. ACM Transaction Autonomic Adaptation System, 1, 2, 223-259.
[2] Fayad, M., & Schmidt, D. C. (1997). Object-Oriented Application Frameworks. Special Issue on Object-Oriented Application Frameworks, 1.
[3] Gorton, I., Liu, A., & Brebner, P. (2003). Rigorous Evaluation Of COTS Middleware Technology. IEEE Computer Society.
[4] Jordan, M., Czajkowski, G., Kouklinski, K., & Skinner, G. (2004). Extending a J2EE Server with Dynamic and Flexible Resource Management. Association for Computing Machinery.
[5] Kephart, J. O., & Chess, D. M. (2003). The Vision Computer. IEEE Computer Society.
[6] Khan, M. J., Awais, M. M., & Shamail, S. (2008). Enabling Self-Configuration in Autonomic Systems using Case-Based Reasoning with Improved Efficiency. Paksitan: IEEE Computer Society.
[7] Kolodner, J. L. (1992). An Introduction to Case-Based Reasoning. Atlanta: College of Computing.
[8] Kusrini, & Luthfi, T. E. (2009). Algoritma Data Mining. Yogyakarta: Andi.
[9] Laddaga, R. (2000). Active software. Proceedings of the International Workshop on Self-Adaptive, 11-26.
[10] Laddaga, R. (1999). Creating Robust Software through Self-Adaptation. IEEE Computer Society.
[11] Liu, Y., & Gorton, I. (2007). Implementing Adaptive Performance Management in Server Applications. Chicago: IEEE Computer Society.