Analisis Performansi Marmoset Untuk Penilaian Pemrograman

Joe Lian Min, Ani Rahmani, dan Bambang Wisnuadhi

Jurusan Teknik Komputer dan Informatika, Politeknik Negeri Bandung,Bandung 40012
E-mail : joelianmin@jtk.polban.ac.id, anirahma@jtk.polban.ac.id, bw@jtk.polban.ac.id

ABSTRAK

Teknologi *autograding* saat ini telah berkembang, khususnya digunakan dalam kompetisi pemrograman. Marmoset merupakan sistem *autograding open source*, yang telah digunakan di Maryland University untuk mendukung pengajaran pemrograman. Penelitian ini dimaksudkan untuk mengetahui performansi konfigurasi *server*, jika Marmoset digunakan untuk menilai sejumlah program dari sejumlah *user* secara otomatis, untuk jenis soal yang didefinisikan. Telaah performansi suatu konfigurasi *server* perlu dilakukan agar implementasi sistem *autograding* dapat stabil untuk mengeksekusi sejumlah program. Diharapkan, dengan mengetahui performansi sebuah konfigurasi, dapat dimanfaatkan untuk menyiapkan instrumen pembelajaran pemrograman secara formal, maupun latihan untuk persiapan kompetisi. Soal untuk uji coba digunakan 3 program, yang tipikal digunakan dalam pengajaran pemrograman dasar. Uji coba dilakukan dengan melibatkan 50-500 *user*. Analisis performansi difokuskan untuk melihat *respon time* pada setiap konfigurasi. *Hardware* untuk uji coba digunakan 1 komputer sebagai *submit server* (untuk menerima) dan 3 komputer *build server* (untuk mengeksekusi). Komputer yang difungsikan sebagai *build server*, dapat menjalankan 2 proses *build server*. Kesimpulan dari penelitian ini adalah terjadi penurunan *respon time* yang besar ketika jumlah *user* bertambah disebabkan oleh *overhead* yang diakibatkan oleh *submit server* dan/atau *build server*. Penurunan performansi terjadi lebih besar pada konfigurasi 6 *build server* dibanding dengan 2 *build server*. Hal ini disebabkan karena *request job* dari *build server* meningkat, sehingga *respon time* dari *submit server* menurun secara signifikan.

Kata Kunci
Teknologi autograding, submit server, build server, pemrograman, analisis performansi, respon time

1. PENDAHULUAN

Aktivitas pemrograman, di samping merupakan aktivitas wajib bagi pelajar / mahasiswa bidang Teknologi Informasi (TI), kini telah memiliki makna "entertain" tersendiri, yang mampu menyedot perhatian masyarakat. Berbagai *event* diselenggarakan dalam rangka menumbuhkan minat masyarakat dalam dunia pemrograman.

Pemrograman merupakan aktifitas intelektual yang kompleks dan merupakan keterampilan utama bagi mahasiswa tahun pertama bidang TI [1]. Memiliki skill motorik dan praktis pemrograman sama pentingnya dengan pemaham-an terhadap konsep, karena pemrograman merupakan keterampilan intelektual yang menun-tut keseimbangan teori dan praktis. Pengajaran pemrograman akan sangat abstrak dan sulit ditangkap jika siswa hanya dihadapkan pada konsep-konsep tanpa pernah bermain dengan komputer dan pemroses bahasanya [2].

Untuk menunjang kegiatan pembelajaran dan latihan pemrograman, diperlukan suatu instrumen yang mendukung, misalnya dengan adanya sejumlah soal latihan yang dapat men-*drill* siswa untuk secara terus menerus dan mandiri berlatih memrogram. Masalahnya adalah latihan dan pekerjaan siswa perlu dievaluasi, karena salah satu aspek pengajaran adalah evaluasi dan penilaian. Suatu proses penilaian yang baik dan juga cepat perlu dilakukan untuk merespon pekerjan siswa, sehingga siswa dapat segera mengetahui kualitas dari program yang telah dibuatnya. Semakin cepat siswa mengetahui 'kesalahan' atau 'kekurangan' dari program yang dibuatnya, semakin cepat pula siswa dapat memperbaikinya, sehingga solusi yang diberikan dapat lebih cepat pula disempurnakan.

Namun terdapat masalah dalam penilaian tugas pemrograman yaitu bahwa proses penilaian bukanlah hal yang mudah dan sederhana, memerlukan waktu yang tidak sebentar, serta menuntut dilakukannya tahapan yang mungkin dapat membosankan yaitu pada saat mengompilasi dan menguji kebenaran dari program tersebut [3], serta memberi umpan balik secepat mungkin terhadap hasil kerja siswa [4].

Sejauh ini, *software* yang dapat digunakan untuk proses evaluasi secara otomatis telah banyak beredar dan telah siap digunakan. Satu permasalahan penting yang diteliti pada penelitian ini adalah sejauh mana performansi sistem *autograding*.

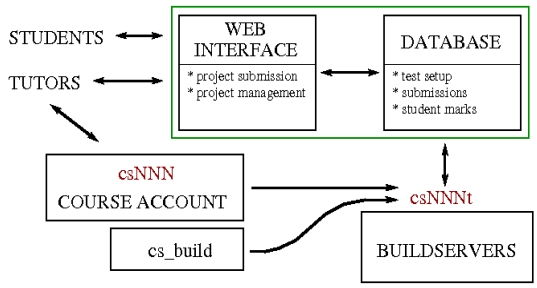
Penelitian ini dilakukan untuk menelaah penerapan *tools autograding* Marmoset dalam mengevaluasi sejumlah program, dengan berbagai sistem konfigurasi *hardware*. Untuk uji coba digunakan soal yang tipikal digunakan dalam pengajaran pemrograman dasar, dan bahasa pemrograman yang digunakan untuk uji coba adalah bahasa Java.

2. SISTEM MARMOSET

Marmoset adalah *framework* yang dapat diguna-kan untuk melakukan penilaian otomatis terhadap program. *Framework* ini juga dapat menyimpan tugas yang telah dikerjakan dan pengajar dapat melihat peningkatan kemampuan siswa.

Marmoset telah digunakan untuk menguji tugas pemrograman siswa dan me-*reviu source code*di University of Maryland. Marmoset dapat digunakan untuk beberapa bahasa pemrograman, dan dirancang untuk bekerja dengan baik dengan skala proyek kecil maupun besar, hingga puluhan ribu baris kode.

Alur kerja Marmoset dapat dilihat pada gambar 1 dan arsitektur Marmoset pada gambar 2. Tujuan dari proyek Marmoset ada dua yaitu untuk meningkatkan pengalaman belajar program untuk siswa dan untuk mempelajari bagaimana siswa belajar program. Dalam memenuhi tujuan tersebut, Marmoset dapat memberikan keuntung-an yakni untuk memotivasi siswa serta untuk tujuan pedagogik bagi pengajar. Secara khusus, Marmoset dapat memberikan siswa dan pengajar umpan balik awal dalam proses belajar, sehingga dapat mendorong siswa untuk menggunakan pengujian untuk menemukan *bug* dalam program mereka. Sistem Marmoset juga dapat menangkap *snapshot* dari pekerjaan siswa setiap kali mereka menyimpan programnya [5].

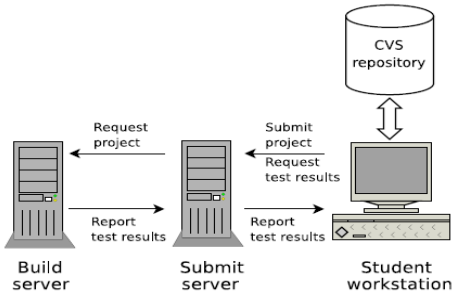


Gambar 1: Mekanisme Marmoset [5]

Kemampuan Marmoset dalam menilai, didasar-kan pada *source code* yang di-*submit* oleh siswa, untuk selanjutnya diuji berdasarkan *input-output* menggunakan program *testcase* yang telah disiapkan

Marmoset dijalankan di dalam sebuah *webserver* dan *servlet container* yang bernama Apache Tomcat. Untuk mengakses sistem Marmoset terbagi menjadi 3 *user*, yaitu **admin**, ***instructor***, dan ***student***.

Marmoset memiliki 2 jenis *server*, yaitu *Submit Server* dan *Build server*. *Submit Server* berfungsi untuk menerima dan menyimpan *Project* atau *source code* yang di-*submit* oleh *student*, sedangkan *Build Server* berfungsi untuk *build Project* dan *test cases* dari *Project* yang akan diujiserta eksekusi *test case(untuk selanjutnya akan disebut job)*. *Build Server* akan secara aktif me-*request* kepada *submit server* sebuah *job*, kemudian mengeksekusi *job* itu serta melaporkan hasil eksekusi *job* ke *submit server*.



Gambar 2: Arsitektur Marmoset [5]

3. METODE

Terdapat 6 kegiatan yang dilaksanakan pada penelitian ini, yaitu penentuan soal untuk studi kasus, eksplorasi teknologi, pendefinisian jumlah *user*, pelaksanaan uji coba, evaluasi terhadap hasil uji coba, dan penarikan kesimpulan.

3.1 Penentuan Program untuk Studi Kasus

Program untuk studi kasus disiapkan 3 buah soal, yang dipilih atas pertimbangan bahwa soal tersebut sering dijadikan bahan latihan dalam pengajaran pemrograman dasar (Tabel 2).

Tabel 2: Karakteristik Program untuk Uji Coba

No Program	Jumlah modul	Jumlah pencabangan	Jenis perulangan	Jumlah baris
1	1	Tidak bersarang 2	tidak bersarang, 2 blok	33
2	1	Bersarang 1 blok	Bersarang, 1 blok	31
3	3	Tidak bersarang 2 blok	Bersarang 3 blok	43

Soal untuk uji coba yang digunakan adalah mencari faktor dari sebuah bilangan (Program-1), membuat deret fibonaci (Program-2), dan studi kasus *sorting* dengan *bubble*, *selection*, dan *insertion sort* (Program-3).

3.2 Eksplorasi Teknologi Marmoset

Teknologi *autograding* yang digunakan adalah Marmoset. Kegiatan eksplorasi dilakukan untuk mengenali kemampuan Marmoset**,** khususnya dalam melakukan penilaian terhadap program**.** Kegiatan eksplorasi, dimulai dari *download software* Marmoset, instalasi *software* pada *server*dan memahami interaksi antara *submit server* dan *build server*.

Karena penelitian difokuskan untuk melihat *respon time* Marmoset sebagai *server* dalam mengeksekusi sejumlah

program, maka dilakukan juga eksplorasi untuk mencoba menggabungkan lebih dari 1 komputer sebagai *server*.

3.3 Pendefinisian Jumlah *User* untuk Eksperimen

Jumlah *user* yang dicobakan dalam eksperimen ini adalah 50, 100, 250, 350, dan 500 *user*. Pada penelitian ini, *user* yang dilibatkan bukan *user* sesungguhnya melainkan sebuah program yang mengemulasikan pengguna sesungguhnya. Penggunaan *program emulasi* ini untuk mengefisienkan uji coba dan dipandang tidak akan memengaruhi kualitas percobaan yang dilakukan.

3.4 Perancangan Konfirgurasi Server

Hardware untuk *server* digunakan *hardware* yang tersedia di Jurusan Teknik Komputer dan Informatika Polban (tidak digunakan di lab). Beberapa komputer dicoba untuk digabungkan sehingga menjadi satu set *server* yang dapat mengeksekusi sejumlah program. Pengamatan dilakukan untuk mengetahui waktu eksekusi (*respon time*) terhadap sejumlah program, pada setiap program.

Konfigurasi *server* disesuaikan dengan ketersediaan *hardware* yang dapat digunakan. Dalam hal ini ada 4 komputer yang dijadikan *server*, yakni 1 untuk *submit server*, dan 3 untuk
build server. Setiap komputer yang berperan sebagai *build server*, difungsikan sebagai 2 *build server* (sistem Marmoset memungkinkan lebih dari 1 *build server* dapat difungsikan dalam 1 komputer)

Dalam pelaksanaan uji coba, *server* yang digunakan memiliki konfigurasi seperti terlihat pada Tabel3. Domain **ss.jtk.polban.ac.id** adalah *submit server*, dan tiga *server* lainnya sebagai *build server* yakni **bs1.jtk.polban.ac.id, bs2.jtk.polban.ac.id**, dan **bs3.jtk.polban.ac.id**.

3. 5 Analisis terhadap Hasil Uji Coba

Pengamatan dilakukan terhadap hasil uji coba. Objek yang diamati adalah waktu yang diperlukan (*respon time*) untuk mengeksekusi seluruh program konfigurasi *server* yang digunakan, untuk setiap kategori soal pada setiap kelompok *user*.

Tabel 3: Daftar Spesifikasi Server yang Digunakan

No	Initial	Processor	RAM	OS	Fungsi
1	SS – (*submit server*)	Intel Pentium Dual CPU 1.60GHz	992.9 MiB	ubuntu 11.04 desktop i386	*submit server*
2	BS1 (*build server1*)	Intel Pentium Dual CPU 1.60GHz	992.9 MiB	ubuntu 11.04 desktop i386	*build server*
3	BS2 (*build server2*)	Intel Pentium Dual CPU 2.80GHz	1.7GiB	ubuntu 11.04 desktop i386	*build server*
4	BS3 (*build server3*)	Intel Pentium Dual CPU 2.80GHz	748MB	ubuntu 11.04 desktop i386	*build server*

4. HASIL DAN DISKUSI

4.1 Hasil Uji Coba

Setiap kali *user* melakukan submit tugas, Marmoset akan mencatatkan waktu submit (ts). Pencatatan waktu akan dilakukan lagi ketika *build server* menyerahkan hasil eksekusi(th) kepada *submit server*. Pada penelitian ini, *respon time* diukur dari selisih dua waktu di atas(th-ts).

Perhitungan waktu rata-rata eksesuksi **n** *user* dihitung dengan merata-ratakan *respon time* dari **n** *user*. Hasil perhitungan waktu rata-rata eksekusi 50-350 *user* untuk program-1, program-2, dan program-3, diperlihatkan pada Tabel 4.

Kolom jumlah *user* menunjukkan jumlah *user* yang berhasil diujicobakan, yaitu dari 50 sampai 350 *user*.

Pada kolom berikutnya, adalah waktu rata-rata eksekusi sebuah job Program-1 ketika menggunakan 2 buah BS. Ketika pengguna berjumlah 50, rata-rata eksekusi Program-1 pada 2 buah BS adalah 102 detik, sedangkan ketika menggunakan 6 buah BS, rata-rata waktu eksekusinya adalah 12 detik. Demikian seterus-nya untuk Program-1, Program-2 dan Program-3. Adapun kolom Rata-rata adalah perhitungan rata-rata eksekusi program-1, program-2 dan program-3 pada 2 BS dan 6 BS.

Pada eksperimen dilakukan juga dengan jumlah *user* sebesar 500. Namun sampai 3 jam, proses tidak selesai (*server* tidak mampu mengeksekusi hingga waktu 3 jam)

4.2 Analisis

Analisis dibagi menjadi 2 bagian, pertama adalah analisis performansi pada saat penambahan *user* dan yang kedua adalah analisis pembandingan performansi antara 2 dan 4 *build server*.

Tabel 4: Waktu Eksekusi Autograding 3 Program pada 2 BS dan 6 BS (detik)

Jumlah user	Program1		Program2		Program3		Rata-rata	
	2 BS	6 BS	2 BS	6 BS	2 BS	6 BS	2 BS	6 BS
50	102	12	103	10	82	10	95.67	10.67
100	150	19	175	20	165	19	163.33	19.33
200	289	41	286	43	209	44	261.33	42.67
350	504	109	504	118	273	72	427.00	99.67

4.2.1 Performansi/Stabilitas ketika Penam-bahan User

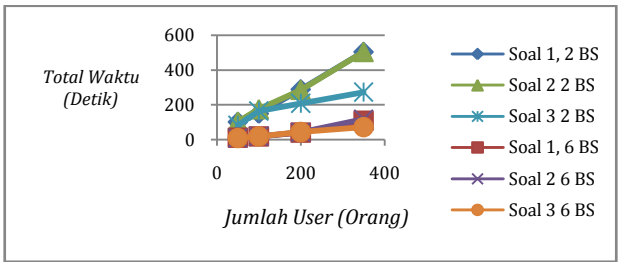
Analisis ini diawali dengan grafik seperti pada gambar 3 dan 4 yang merupakan representasi tabel 4 dalam bentuk grafik. Waktu eksekusi program ketika dijalankan secara *stand-alone* adalah sekitar 1.5 detik untuk masing-masing

program. Analisis waktu eksekusi 50 buah program-1 dengan 6 BS adalah : setiap BS akan mengerjakan 8-9 *jobs*. Dari tabel 4 dapat dilihat waktu rata-rata eksekusinya adalah 12 detik, sehingga waktu total tiap BS bekerja adalah 8 job * 12 detik/job = 96 detik. Waktu proses eksekusi antrian dengan n *job* dan tiap *job*-nya memerlukan waktu t merupakan perhitungan jumlah suku ke-n dari deret aritmatika dengan nilai awal t dan beda t juga. Rumus dari jumlah suku ke-n deret aritmatika adalah 1/2n(2a+(n-1)b); n:jumlah suku, a: nilai awal dan b: beda. Dengan mengaplikasikan rumus ini, didapat nilai waktu proses tiap *job* sebesar 2.64 detik. Terlihat ada waktu *overhead* sebesar 1.14 detik. Dengan cara perhitungan yang sama, ketika 350 *job* diselesaikan dengan 6 BS, diperoleh waktu *overhead* sebesar 2.1 detik. Jika dibandingkan antara 50 job dan 350 job, ada kenaikan waktu *overhead* sebesar 90%. Jika sistem stabil, waktu *overhead* tidak akan berubah banyak. Waktu *overhead* ini adalah kontribusi dari submit *server* dan/atau *build server*.

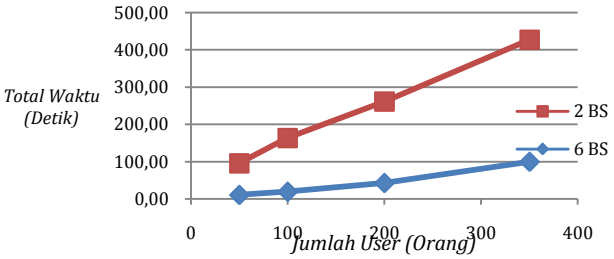
Ketidakstabilan muncul ketika jumlah *user* 500, ini ditandai dengan tidak dapat diselesaikannya pengolahan hingga waktu telah lebih dari 3 jam.

4.2.2 Perbandingan Performansi 2 dan 6 *Build Server*

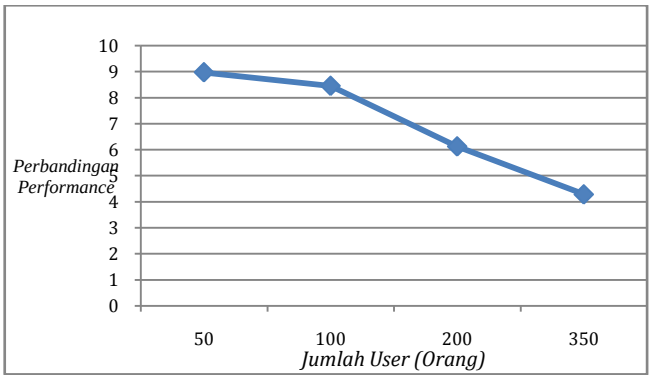
Pada gambar 5 ditunjukkan perbandingan rata-rata *respon time* 3 program antara 2 *build server* dan 6 *build server*. Seperti telah dijelaskan sebelumnya, tiap komputer *build server* menjalankan 2 buah proses *build server*. Pertama akan dibahas perbandingan potensi power yang dimiliki oleh masing-masing konfigurasi. Komputer-komputer yang digunakan pada eksperimen ini adalah Intel Pentium Dual Core dengan 2 jenis *clock*, yaitu 1.6GHz dan 2.8GHz dan besaran RAM yang berbeda-beda namun semuanya lebih besar dari 512MByte. Mengingat 2 buah *build server* memakan sekitar 100Mbyte, perbedaan besaran RAM diabaikan. Pada konfigurasi 2 *build server*, komputer yang digunakan adalah 1 komputer yang 1.6 GHz, sedangkan pada konfigurasi 6 build server adalah 1 komputer yang 1.6 GHz dan 2 komputer yang 2.8 GHz. Pembandingan potensi *power* masing-masing core yg 2.8GHz dan 1.6GHz adalah 1.75. Dengan pendekatan bahwa 1.6GHz jadi satuan, perbandingan potensi *power* 6 BS dan 2 BS adalah 1.75 * 2 core/computer *2komputer + 1*2 core = 9. Pada gambar 5 dapat dilihat ketika jumlah *user* 50, perbandingan antara 6 BS dan 2 BS adalah 8.9 (mendekati hitungan ideal). Namun seiring dengan meningkatnya jumlah *user*, terjadi penurunan yang cukup tajam. Ini sejalan dengan analisis sebelumnya, yaitu pada 6 BS antara jumlah pengguna 50 dan 350 terjadi peningkatan *overhead* sebesar 90%. Penyebab dari penurunan ini adalah di sisi *submit server*. Ketika *request job* dari *build server* meningkat, terjadi penurunan *respon time* yang signifikan dari *submit server*.



Gambar 3: Waktu Eksekusi Soal1-Soal3 pada 2 BS dan 6 BS



Gambar 4: Rata-rata Waktu Eksekusi 3 Soal pada 2 BS dan 6 BS



Gambar 5: Perbandingan Performansi antara 2 BS dan 6 BS

5. KESIMPULAN

Dari pembahasan di atas, dapat disimpulkan beberapa hal berikut:

1. Penurunan *respon time* yang besar ketika jumlah pengguna bertambah disebabkan oleh *overhead* yang diakibatkan oleh *submit server* dan/atau *build server*.
2. Terjadi penurunan performansi yang lebih besar pada konfigurasi 6 *build server* dibandingkan dengan 2 *build server*. Penyebab dari penurunan ini adalah *request job* dari *build server* meningkat, sehingga *respon time* dari *submit server* menurun secara signifikan.

UCAPAN TERIMA KASIH

Ucapan terima kasih, kami sampaikan kepada mahasiswa Jurusan Teknik Komputer dan Informatika - Polban, yang telah bersama-sama mengeksplorasi Marmoset, sehingga penelitian dapat berjalan dengan baik.

DAFTAR PUSTAKA

[1] Troug,N., Roe,P., dan Peter Bancroft,P, "Static Analysis of Students Java Program", Proceeding. The Sixth Australian Computing Education Conference (ACE2004)
[2] Liem, I, "Aspek Pedagogi Pengajaran Pemrograman", Depdiknas RI, 2004.
[3] Patil, A, "Automatic Grading of Programming Assignments." Master's Projects. Paper 51. San Jose State University, 2010.
[4] Harris, JA., Elizabeth S.Adams,. dan Harris, NA," Making Program Grading Easier (but not Totally Automatic)". A paper at James Madison University, 2003.
[5] Spaco, J, N. Padua-Perez, F. Emad, J.k. Hollingsworth, W. Pug, dan D. Hovemever, "Experiences with Marmoset", University of Maryland, 2005.