Jurnal TEKNOINFO, Vol. 14, No. 1, 2020, 22-26, ISSN: 2615-224X
DOI : 10.33365/jti.v14i1.466

ANALISIS METODE *LOAD BALANCING* DALAM MENINGKATKAN KINERJA *WEBSITE E-LEARNING*

Sampurna Dadi Riskiono[1)], Donaya Pasha[2)]

[1]Teknik Elektro, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia
[2]Teknologi Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia
[1,2]Jl. Z. A. Pagar Alam No.9 -11, Labuhan Ratu, Kedaton, Kota Bandar Lampung, Lampung
Email: [1] sampurna.go@teknokrat.ac.id, [2] donayapasha@gmail.com

Abstrak

Saat ini, dunia pendidikan berkembang sangat pesat hal tersebut ditandai dengan penggunaan teknologi informasi sebagai pendukungnnya. Penggunaan teknologi informasi ini menyebabkan proses belajar mengajar menjadi lebih interaktif dan menarik. Diawal, penggunaan teknologi informasi hanya sebatas untuk menyampaikan presentasi materi dengan menggunakan Power Point, Adobe Flash, maupun aplikasi khusus lain yang memiliki fungsi yang sama. Namun, seiring dengan tumbuhnya internet, proses belajar mengajar menjadi banyak memanfaatkan aplikasi yang berbasis internet, diantara penggunaan e-learning. Ketika jumlah pengguna yang mengakses layanan e-learning meningkat dan server tidak dapat mengatasinya, tentu ini akan menjadi masalah. Oleh karenanya diperlukan sistem server yang dapat menangani banyaknya permintaan layanan yang masuk agar skalbilitas dari server e-learning dapat meningkat. Salah satu solusi dari permasalahan tersebut adalah dengan peerapan load balancing. Dalam makalah ini, akan dilakukan evaluasi antara penggunaan server tunggal dan server jamak. Hasil pengujian menunjukkan implementasi dari load balancing memiliki nilai response time 36,4 ms lebih kecil dibandingkan server tunggal yang memiliki waktu response time 51,1 ms pada uji koneksi 500/10 sec. Alhasil dari pengujian yang dilakukan, penerapan load balancing lebih baik dari segi nilai response time jika dibandingkan dengan server tunggal untuk setiap rentang pengujiannya.

Kata Kunci: *load balancing, server E-Lerning, Beban Koneksi, response time, skalabilitas*

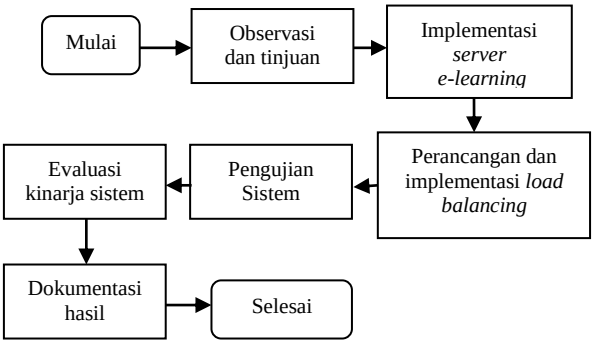
1. Pendahuluan

Seiring dengan semakin kompleksnya layanan dan aplikasi web dalam berbagai bidang, maka permintaan layanan web dari pengguna semakin meningkat. Contoh layanan dan aplikasi web yang populer adalah layanan dan aplikasi bisnis (*e-business*), pendidikan (*e-learning*), berita (-news), dan lain-lain[1]. Dalam meningkatkan sistem layanan *e-learning*, dibutuhkan suatu sistem *server* yang dapat mengatasi sejumlah akses yang tinggi. Hal ini terkait dengan skalabilitas sistem, dimana model pelayanan *server* jamak menjadi sebuah pilihan dengan menerapkan metode *load balancing*. Load balancing pada *server* merupakan salah satu cara yang dapat digunakan untuk meningkatkan kinerja dan ketersediaan *server*, yaitu dengan membagi permintaan layanan yang datang ke beberapa *server* sekaligus, sehingga beban yang ditanggung oleh masing-masing *server* lebih sedikit [2]. Dalam menanggulangi peningkatan kinerja ketika adanya akses yang begitu banyak, maka penggunaan banyak *server* dapat menjadi solusi untuk mengatasi hal tersebut. Himpunan dari banyak *server* disebut dengan kluster *server*. Sehingga dengan menerapkan kluster *server* maka dapat meningkatkan keandalan dan ketersediaan aplikasi [3]. *Server load balancing* memiliki tugas dalam mendistribusikan beban kerja ke banyak *server* dengan mempertimbangkan kapasitas dari setiap *server* yang ada sehingga dapat mengurangi terjadinya kegagalan *server* [4]. Dengan penerapan banyak *server*, maka ketika terjadi kegagalan pada salah satu *server*, layanan yang ada masih terus dapat berjalan. Hal lain adalah beban yang semakin kecil, karena beban yang ada kemudian didistribusikan ke masing-masing *server* sehingga sistem yang ada menjadi terdistribusi. Sistem terdistribusi menyediakan berbagai sumberdaya, sebagai salah satu keuntungan utamanya adalah dapat menyediakan kinerja yang lebih baik serta keandalan dari pada sistem tradisional yang lain dalam kondisi yang sama [5]. Penggunaan banyak *server* untuk sistem terdistribusi membutuhkan metode untuk mengatur pembagian beban secara adil atau merata pada masing-masing *server*. Banyak penelitian telah dilakukan terkait penerapan metode *load balancing* untuk mengatur pembagian beban kerja pada *server* dengan tujuan untuk meningkatkan kinerja sistem. Penerapan load balancing dalam web *server* sangat penting dan dapat merupakan sebuah solusi yang tepat dan efektif untuk menangani beban *server* yang sibuk serta diharapkan dapat meningkatkan skalabilitas pada sistem terdistribusi [6]. Web *server* harus bebas dari

kegagalan baik akibat dari *hardware* ataupun *software*. Dalam kondisi yang lain *fault tolerance* dapat diterapkan untuk mendeteksi dan mentoleransi kerusakan secara *real-time* pada sistem terdistribusi [7].

2. Metode

Dalam penelitian ini, metode *load balancing* akan digunakan untuk meningkatkan kinerja dari *server e-learning* dengan membandingkan kondisi sebelum dan setelah *load balancing* diimplementasikan. Untuk tahap awal akan dilakukan tinjauan pustaka dan observasi. Proses selanjutnya adalah implemantasi sistem e-learning dan dilanjutkan dengan implementasi dari metode *load balancing*. Pengujian dilakukan dengan cara memberikan sejumlah koneksi terhadap *server* e-learning, baik tunggal maupun jamak. Selanjutnya akan dilakukan evaluasi terhadap implementasi metode *load balancing* tersebut dengan menggunkan parameter pengujian yang telah ditentukan. Dari hasil evaluasi, maka akan terlihat nilai hasil dari uji parameter yang telah dilakukan. Untuk memperjelas jalannya penelitian ini secara lebih detail metode yang akan dilakukan ditunjukkan seperti pada Gambar 1.



Gambar 1. Metode Penelitian

Adapun yang menjadi landasan terori dari penelitian ini adalah sebagai berikut :

1. *E-Learning*

E-Learning merupakan kependekan dari *electronic learning* adalah sebuah proses pembelajaran dimana penyampaian materi, diskusi, ujian dan lain-lain yang berkaitan dengan kegiatan perkuliahan dilakukan melalui media elektronik [8]. Dari beberapa sistem e-Learning yang ada, secara umum dapat dibagi berdasarkan sifat interaktivitasnya dan dapat dibedakan kedalam dua kelompok yaitu *e-Learning* statis dan *e-Learning* dinamis. Sistem *e-Learning* dikatakan bersifat statis jika antara pengguna sistem tidak dapat saling berinteraksi, pembelajar hanya dapat melakukan proses *download* bahan-bahan yang diperlukan dan admin hanya dapat melaukan proses *upload file-file* materi. Sistem ini biasanya digunakan hanya sebagai penunjang aktifitas belajar-mengajar yang dilakukan secara tatap muka dikelas. Sedangkan sistem *e-Learning* dapat digolongkan kedalam *e-Learning* yang bersifat dinamis apabila siswa mampu belajar dengan dalam lingkungan yang tidak jauh berbeda dengan suasana kelas dimana di dalam sistem ini terdapat kemungkinan untuk berinteraksi antara pembelajar dan tutornya baik melalui email, *chatting* maupun sarana komunikasi lainnya [9].

2. *Load Balancing*

Load balancing merupakan metodologi jaringan komputer yang bekerja dengan cara mendistribusikan beban kerja dibeberapa komputer atau kluster komputer untuk mencapai pemamfaatan optimal dari sumber daya, memaksimalkan *throughput*, meminimalkan waktu respon, dan menghindari kelebihan beban. Kluster *server* sendiri merupakan gabungan beberapa perangkat komputer yang saling terhubung dan bekerjasama sehingga mereka dapat dilihat sebagai satu sistem dalam banyak aspek, dan kluster komputer biasanya digunakan untuk meningkatkan kinerja dan ketersediaan atas satu komputer [6]. *Load balancer* dapat melakukan berbagai fungsi seperti peyeimbang beban, *traffic engineering*, dan pemindahan jalur trafik. *Load balancer* dapat melakukan pemeriksaan kesehatan pada *server*, aplikasi, dan konten untuk meningkatkan ketersediaan layanan dan pengelolaanya [10]. Dalam penerapan *load balancer* ada berbagai algoritma penjadwalan yang dapat digunakan untuk mendistribusikan beban kepada setiap *server* didalam sekumpulan *server*. Algoritma penjadwalan yang umumnya digunakan adalah :

1. *Round robin*

Untuk model penjadwalan tipe *round robin*, *load balancer* mendistribusikan permintaan layanan sama rata ke seluruh real *server* tanpa memperdulikan kapasitas *server* atau pun beban permintaan layanan. Jika ada empat *real server* (A,B,C,D), maka permintaan layanan 1 akan diberikan *load balancer* kepada *server* A, permintaan layanan 2 ke *server* B, permintaan layanan 3 ke *server* C, permintaan layanan 4 ke *server* D dan permintaan layanan 4 akan kembali ke *server* A. Mekanisme ini dapat dilakukan jika seluruh real *server* menggunakan spesifikasi komputer yang sama. Konsep dasar dari algoritma ini adalah dengan menggunakan time-sharing. Pada dasarnya algoritma ini sama dengan FCFS, hanya saja bersifat *preemptive*. Setiap proses mendapatkan waktu CPU yang disebut dengan waktu quantum (quantum time) untuk membatasi waktu proses, biasanya 1-100 milidetik. Setelah waktu habis, proses ditunda dan ditambahkan pada *ready queue*. Algoritma roundrobin merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat load balancing. Algoritma ini membagi beban secara bergiliran dan berurutan dari satu *server* ke *server* lain sehingga membentuk putaran.

Penjadwalan ini merupakan:

1. Penjadwalan preemptive, bukan di-preempt oleh proses lain, tapi terutama oleh penjadwal berdasarkan lama waktu berjalannya proses, disebut preempt by time.
2. Penjadwal tanpa prioritas. Semua proses dianggap penting dan diberi sejumlah waktu proses yang disebut kwanta (quantum) atau time slice dimana proses itu berjalan.

Ketentuan algoritma roundrobin adalah sebagai berikut:

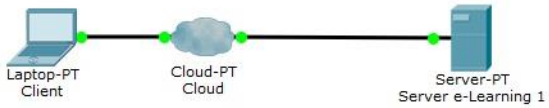
1. Jika kwanta dan proses belum selesai maka proses menjadi runnable dan pemroses dialihkan ke proses lain.
2. Jika kwanta belum habis dan proses menunggu suatu kejadian (selesainya operasi I/O), maka proses menjadi *blocked* dan pemroses dialihkan ke proses lain.
3. Jika kwanta belum habis tapi proses telah selesai, maka proses diakhiri dan pemroses dialihkan ke proses lain. Algoritma *round robin* menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar *quantum time*. Jika *quantum time*-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 *quantum*. Algoritma ini sepenuhnya bergantung besarnya time quantum. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma first come first served. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang. Permasalahan utama pada *round robin* adalah menentukan besarnya *time quantum*. Jika time quantum yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesai dalam 1 *quantum*. Hal ini tidak baik karena akan terjadi banyak switch, padahal CPU memerlukan waktu untuk beralih dari suatu proses ke proses lain (disebut dengan *context switches time*). Sebaliknya, jika time quantum terlalu besar, algoritma Roundrobin akan berjalan seperti algoritma *first come first served*. Time quantum yang ideal adalah jika 80% dari total proses memiliki CPU *burst time* yang lebih kecil dari 1 time quantum [11].

2. *Least connection*

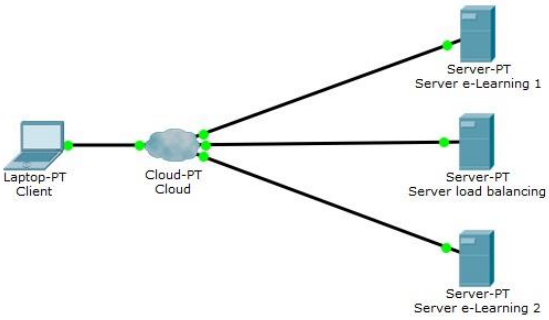
Algoritma ini melibatkan pengiriman permintaan layanan baru ke server dengan jumlah koneksi bersamaan. Metode ini memerlukan penyeimbang beban untuk melacak jumlah koneksi aktif secara bersamaan pada setiap server dan mengirim permintaan ke server dengan sedikitnya jumlah koneksi yang aktif bersamaan [11].

3. Pembahasan

Pada Gambar 1 menjelaskan bagaimana bentuk dari arsitektur server tunggal yang mana hanya terdiri dari *single server* . Selanjutnya pada Gambar 2 merupakan arsitektur server jamak yang akan diterapkan dalam penelitian ini. Yang mana terdiri dari *server load balancer*, serta dau buah *server e-learning*.



Gambar 1. Arsitektur *Server* Tunggal



Gambar 2. Arsitektur *server* jamak dengan *Load Balancing*

Pada Gambar 1 menunjukkan bahwa *server load balancer*, App *server*1 *e-learning*, App *server*2 *e-learning* terletak pada jaringan *cloud*. Sedangkan client dapat melakukan akses melalui jaringan publik. Untuk detail dari antarmuka jaringan yang terpasang akan diperlihatkan pada

Tabel 1. Nama *Server* dab IP Address

Nama *Server*	**IP Address**
Server Load Balancer	103.41.205.217
Server 1 E-Learning	103.41.204.218
Server 1 E-Learning	103.41.205.204
Client	192.168.7.131

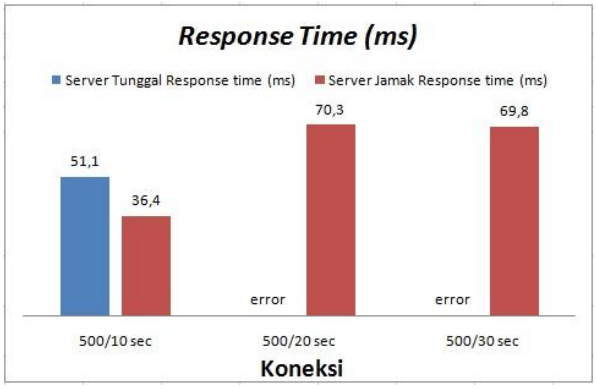
Pada rincian di atas, menunjukkan bahwa alamat IP 103.41.205.217 merupakan alamat yang dimiliki oleh *load balancer* yang berfungsi untuk meneruskan permintaan ke aplikasi *server*. Ketika pengguna meminta konten di *server* aplikasi *e-learning*, maka semua permintaan akan melewati *server load balancer*. Jadi pengguna tidak akan mengetahui *real server* mana yang melayani. Kemudian permintaan layanan tersebut akan dibagikan oleh *server load balancer* kepada *real server* yang terdaftar pada *load balancer* dengan menyesuaikan pada algoritma penjadwalan yang telah disematkan.

Selanjutnya setiap *real server* akan terhubung ke *server* database untuk menyesuaikan data yang akan digunakan, setelah itu permintaan akan di berikan kepada client melalui *server load balancer* kembali. Pengamatan dalam penelitian ini dilakukan dengan mengambil dua parameter yaitu *throughput* dan *response time*. Pengujian ini bertujuan untuk melakukan evaluasi kinerja dengan diimplementasikannya metode *load balancing*. Dalam hal ini yang mana pada sisi klien akan dilakukan permintaan secara bertahap melalui *tool* httperf untuk melihat throughput dan response time. Untuk lebih jelasnya terlihat pada Tabel 2 yang merupkan hasil rata-rata dari permintaan layanan yang di lakukan oleh client melalui *tool* httperf. Dimana permintaan layanan tersebut akan dilakukan secara bertahap. Dimulai dari jumlah koneksi 500/10 *sec*, 500/20 *sec*, dan 500/30 *sec* yang dilakukan koneksi ke *server* secara bersamaan dalam satu waktu. Dari sini maka dapat dilakukan pengamatan serta membandikan kinerja dari kedua model yang telah di implementasikan. Berdasarkan pada hasil yang telah diperoleh dari nilai *response time*.

Tabel 2. Nama *Server* dab IP Address

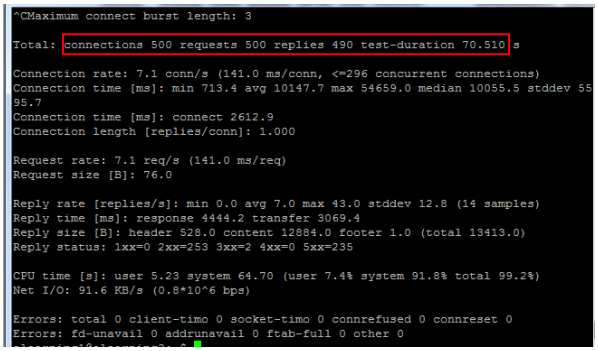
Koneksi/*sec*	Server Tunggal *Response time (ms)*	Server Jamak *Response time (ms)*
500/10 *sec*	51,1	36,4
500/20 *sec*	Error	70,3
500/30 *sec*	Error	69,8

Dari Gambar 3 memperlihatkan perbandingan secara grafik antara sistem server tunggal dan sistem server jamak dari nilai *response time* yang telah dikumpulkan.



Gambar 3. Grafik *Response Time (ms)*

Hasil dari perbandingan kedua model arsitektur baik server tunggal dan server jamak dapat dilihat pada Tabel 2, dimana penerapan *load balancing* memiliki nilai *response time* lebih kecil di bandingkan dengan penerapan server tunggal dari setiap rentang pengujinnya. Pada uji koneksi 500/10 *sec* *nilai reponse time* dari server jamak adalah 36,4 ms, lebih kecil dibandingkan penerapan server tunggal yang memiliki nilai *response time* sebesar 51, 1 ms. Ini menujukan bahwa kecepatan response time yang didukung penerapan load balancing cepat dibandingkan hanya dengan penerapan server tunggal. Sehingga dari segi layananan, penerapan load balancing layak untuk diimplemantasikan karena ini merupakan bagian dari *quality of service* (QoS) dari suatu layanan. Hal serupa juga terjadi pada uji koneksi selanjutnya, dimana pada uji koneksi 500/20 *sec*, dan 500/30 *sec* penerapan server jamak yang didukung metode *load balancing* lebih unggul bila dibandingkan dengan penerapan server tunggal. Bahkan untuk nilai koneksi 500/20 *sec*, dan 500/30 *sec*, server tunggal mengalami *error* koneksi, dimana tidak semua layanan dapat diterima dan diteruskan kembali. Hal tersebut dapat dilihat dari pantauan saat dilakukan uji koneksi tersebut seperti diperlihatkan pada Gambar 4.



Gambar 3. Hasil uji koneksi terhadap server tunggal

Dalam penelitian ini, *load balancer* masih bersifat tunggal. Belum dikembangkan bagaimana membangun sebuah sistem yang memiliki ketersediaan yang tinggi dengan menerapkan sistem yang *fault tolerance*. Dengan sistem yang *fault tolerance* diharapkan ketersediaan sistem akan terus terjaga.

4. Kesimpulan

Dari hasil pengujian yang telah dilakukan maka dapat diambil kesimpulan, yaitu :

1. Hasil perbandingan nilai rata-rata menunjukkan least connections memiliki nilai response time 50.27 ms lebih kecil dibandingkan round robin yang memiliki waktu respons 50.88 ms. Ini menunjukkan respon least connections lebih cepat dibanding round robin. Untuk throughput, least connection juga memiliki nilai yang lebih besar yaitu 29.51 Kb/sec dibandingkan round robin yang memiliki nilai 29.21 Kb/sec. Ini menunjukkan bahwa jumlah paket yang dapat dilewatkan least connections lebih besar dari pada round robin.

2. Alhasil, dari pengujian yang telah dilakukan, menunjukkan penerapan *load balancing* dalam

menikatkan kinerja telah berhasil diimplementasikan. Dengan melihat nilai *response time* yang lebih kecil disbanding dengan penggunaan server tunggal.

Ucapan Terimakasih

Ucapan TerimakasihTerima kasih kepada Direktorat Riset dan Pengabdian kepada Masyarakat (DRPM) Dikti yang telah mendanai kegiatan penelitian ini yakni pada skema Penelitian Dosen Pemula (PDP) sesuai dengan SK Penetapan Pemenang Hibah PDP nomor: T/140/E3/RA.00/2019 tanggal 28 Februari 2019 dan Kontrak Pelaksanaan Penelitian Nomor : 010/LPPM-UTI/FTIK/PDP-MONO/V/2019 tanggal 2 Mei 2019. Terima kasih juga peneliti sampaikan kepada LPPM Universitas Teknokrat Indonesia yang telah memfasilitasi kegiatan penelitian ini.

Daftar Pustaka

[1] N. Angsar, "Pengujian Distribusi Beban Web dengan Algoritma Least Connection dan Weighted Least Connection," *Jnteti*, vol. 3, no. 1, pp. 24–28, 2014.

[2] D. Lukitasari, F. Oklilas, F. I. Komputer, and U. Sriwijaya, "Analisis Perbandingan Load Balancing Web Server Tunggal Dengan Web server Cluster Menggunakan Linux Virtual Server," vol. 5, no. 2, pp. 31–34, 2010.

[3] S. D. Riskiono, S. Sulistyo, and T. B. Adji, "Kinerja Metode Load Balancing dan Fault Tolerance Pada Server Aplikasi Chat," *Pros. Semin. Nas. ReTII*, 2017.

[4] C. Kopparapu, "Load Balancing Servers , Firewalls , and Caches Chandra Kopparapu."

[5] S. Raharjo, "Implementasi Load Balancing Web Server Menggunakan Metode LVS-NAT," vol. 8, no. 2, 2012.

[6] U. Haluoleo, K. Bumi, and T. Anduonohu, "Peningkatan Kinerja Siakad Menggunakan Metode Load Balancing dan Fault Tolerance Di Jaringan Kampus Universitas Halu Oleo," vol. 10, no. 1, pp. 11–22, 2016.

[7] H. Chang, "Load Balancing and Fault-Tolerance for Scalable Network File Systems Using by Web Services," pp. 351–356.

[8] A. Ambarita, "Implementasi Sistem E-Learning Menggunakan Software Moodle Pada Politeknik Sains Dan Teknologi Wiratama Maluku Utara," *IJIS - Indones. J. Inf. Syst.*, vol. 1, no. 2, 2017.

[9] B. Suteja, U. K. Maranatha, A. Harjoko, and U. G. Mada, "User Interface Design for e-Learning System User Interface Design for e-Learning System," no. June, 2015.

[10] S. Malik, "Dynamic load balancing in a network of workstations," *Pap. Parallel Process. Course, Carlet. …*, no. 219762, 2000.

[11] G. Triono, "Implementasi Load Balancing Dengan Menggunakan Algoritma Round Robin Pada Kasus Pendaftaran Siswa Baru Sekolah Menengah Pertama Labschool Unesa Surabaya," pp. 169–176, 2015.