# Kelayakan Raspberry Pi sebagai Web Server: Perbandingan Kinerja Nginx, Apache, dan Lighttpd pada Platform Raspberry Pi

Rahmad Dawood[1,2], Said Fairuz Qiana[1], dan Sayed Muchallil[1]
[1)] Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala
[2)] Telematics Research Center, Universitas Syiah Kuala
Jl. Tgk. Syech Abdurrauf, Banda Aceh 23111
e-mail: rahmad.dawood@unsyiah.ac.id

***Abstrak*—Raspberry Pi merupakan komputer mini yang dapat difungsikan sebagai PC pada umumnya. Karena dapat difungsikan sebagai PC maka Raspberry Pi juga dimungkinkan untuk menjalankan aplikasi *web server*. Tulisan ini akan memaparkan hasil pengujian kelayakan dan kinerja aplikasi web server yang dijalankan pada Raspberry Pi. Pengujian dilakukan terhadap tiga aplikasi *web server* yang paling populer saat ini, yaitu: Apache, Nginx, dan Lighttpd. Dengan memakai parameter kinerja: *maximum request* dan *reply time*. Hasil pengujian menunjukkan bahwa Raspberry Pi layak untuk menjalankan kesemua aplikasi *web server* yang diuji namum kinerja terbaik diberikan oleh Nginx yang diikuti oleh Lighttpd dan Apache.**

**Kata kunci: *Raspberry Pi, web server, Apache, Lighttpd, Nginx, web server performance***

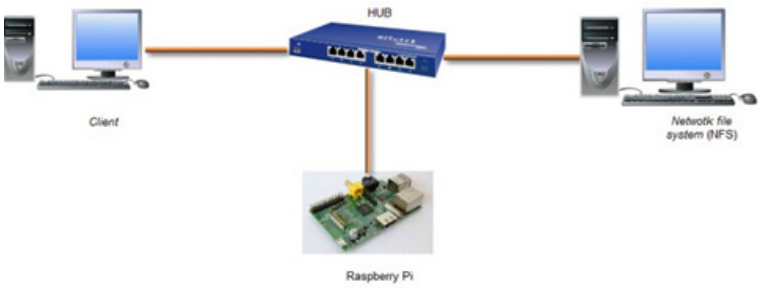
***Abstract*—Raspberry Pi is a small-sized computer, but it can function like an ordinary computer. Because it can function like a regular PC then it is also possible to run a web server application on the Raspberry Pi. This paper will report results from testing the feasibility and performance of running a web server on the Raspberry Pi. The test was conducted on the current top three most popular web servers, which are: Apache, Nginx, and Lighttpd. The parameters used to evaluate the feasibility and performance of these web servers were: maximum request and reply time. The results from the test showed that it is feasible to run all three web servers on the Raspberry Pi but Nginx gave the best performance followed by Lighttpd and Apache.**

**Keywords: *Raspberry Pi, web server, Apache, Lighttpd, Nginx, web server performance***

## I. PENDAHULUAN

Aplikasi *web server* cenderung dijalankan pada *hardware* yang berspesifikasi tinggi. Kecendrungan ini dikarenakan adanya ekspektasi bakal banyak permintaan yang harus ditangani secara bersamaan oleh aplikasi *web server* tersebut sehingga memerlukan *hardware* yang relatif besar agar dapat melayani dengan baik. Sebagai contoh, spesifikasi *hardware* Dell PowerEdge R620 E5-2690 yang lazim dipergunakan untuk menjalankan aplikasi *web server* adalah sebagai berikut: prosesor Intel Xeon E5-2609 2.40GHz, memori 8GB (2x4GB), *harddisk* RAID dengan delapan buah HDD, dan memakai daya 1100W [1] serta berdimensi 1U (19" x 1,75" x 17,7") dan memiliki berat 18,58kg [2]. Tetapi di lapangan kebanyakan aplikasi *web server* tidak akan menangani sebegitu banyak permintaan sehingga bakal tidak akan memanfaatkan sepenuhnya potensi dari *hardware* yang dimilikinya. *Hardware* dengan spesifikasi demikian harus ditempatkan pada suatu ruang server khusus dan kurang sesuai untuk pemakaian yang bersifat *mobile* atau *semi-mobile*. Misalnya: pada sistem pemantauan lingkungan yang ditempatkan pada lokasi-lokasi terpencil (e.g. di tengah hutan) [3] dimana data pantauannya hanya diakses oleh suatu sistem lain agar dapat diarsipkan dan dianalisa lebih lanjut oleh para penelitinya; pada sistem pengukuran di daerah-daerah ekstrim (e.g. daerah kutup) [4] dimana hasil pengukurannya perlu diakses secepat mungkin oleh para peneliti yang terlibat baik yang berada di lokasi pengukuran maupun yang berada di lokasi lain; atau pada sistem yang dipakai oleh tim asesmen suatu daerah bencana [5] dimana hasil asesmennya perlu segera diakses oleh sejumlah pihak yang berkepentingan dengan bencana tersebut. Dari ketiga contoh sistem ini penempatan *hardware* jelas tidak bisa pada ruang khusus tetapi harus dibawa ke lokasi pemakaian dan permintaan yang harus ditanganipun tidak terlalu besar sehingga *hardware* dengan spesifikasi di atas akan tidak sesuai.

Salah satu alternatif untuk menanggulangi kebutuhan akan spesifikasi *hardware* yang tinggi ini adalah dengan memanfaatkan Raspberry Pi. Raspberry Pi merupakan komputer mini yang memiliki ukuran kecil yaitu sebesar kartu ATM tetapi mampu menjalankan tugas yang sama dengan komputer PC. Spesifikasi teknis Raspberry Pi terdiri atas: prosesor Broadcom BCM2835 700MHz, memori 512MB untuk Tipe B atau 256MB untuk tipe A, memakai SD Card sebagai pengganti *harddisk*, dan memakai daya 2,5W serta berdimensi 3,37" x 2,21" x




Gambar 1.Topologi alternatif web server pada perangkat Raspberry Pi

0.83" dan memiliki berat 45gr [6],[7].

Karena Raspberry Pi merupakan sebuah komputer mini maka dimungkinkan juga untuk menjalankan aplikasi *web server* [7]. Namun demikian kinerja aplikasi *web server* pada Raspberry Pi masih belum diketahui padahal informasi ini penting agar dapat ditarik kesimpulan apakah *hardware* ini layak untuk menjalankan aplikasi *web server*. Ditambah lagi, saat ini terdapat lebih dari 60 aplikasi *web server* [8] sehingga perlu dikaji aplikasi yang mana yang paling cocok untuk dijalankan pada Raspberry Pi.

## II. Latar Belakang

Raspberry Pi merupakan hasil karya sebuah yayasan nirlaba asal Inggris yaitu Raspberry Pi Foundation yang awalnya membuat perangkat ini untuk menarik minat anak-anak agar bercita-cita menjadi seorang pengembang atau *developer* baik di bidang *hardware* maupun software [7]. Raspberry Pi dirilis dengan lisensi *Open-Source Hardware* yang berarti rancangan perangkat kerasnya dirilis ke publik agar dapat bebas dipelajari, dimodifikasi, didistribusi, dirakit, dan dijual sesuai rancangan aslinya [9]. Karena dirilis dengan lisensi *Open-Source Hardware*, Raspberry Pi dengan seketika menjadi populer dan telah dipergunakan untuk berbagai keperluan, diantaranya untuk *hardware* yang menjalankan: media center, *networked computer*, dan aplikasi *web server* [7]. Penelitian ini mengkaji kelayakan Raspberry Pi sebagai *hardware* untuk menjalankan aplikasi *web server* dengan cara mengukur kinerja aplikasi tersebut saat dijalankan pada Raspberry Pi.

Aplikasi *web server* merupakan suatu perangkat lunak yang berjalan di sisi server dan bertugas untuk menerima permintaan dari *web browser*, menerjemahkan permintaan tersebut, dan mengembalikan ke *web browser* hasil dari permintaan itu [10]. Permintaan dari *web browser* dapat berupa permintaan untuk mengirimkan suatu halaman statis (sebuah berkas yang terdapat di server) atau permintaan untuk suatu halaman dinamis (halaman yang dibangkitkan oleh suatu program yang berjalan di server). Jika permintaan adalah untuk suatu halaman statis maka aplikasi *web server* akan mencari berkas tersebut dan mengirimkannya ke *web browser*. Sedangkan jika permintaan adalah untuk suatu halaman dinamis maka aplikasi *web server* akan memanggil program yang sesuai, menampung keluaran dari program tersebut, dan mengirimkan hasil keluaran tersebut ke *web browser* yang memintanya. Dalam penelitian ini, kinerja aplikasi *web server* yang dijalankan pada Raspberry Pi diukur saat melayani kedua tipe halaman ini.

Hasil *survey* W3Techs per Juni 2014 [8] mengidentifikasi adanya lebih dari 60 aplikasi *web server* yang saat itu dipakai di Internet. *Survey* ini juga mengidentifikasi sepuluh *web server* yang paling banyak dipakai, yaitu: Apache, Nginx, Microsoft IIS, LiteSpeed, Google Servers (*web server internal Google*), Tomcat (berbasis Apache), Lighttpd, Yahoo Traffic Server (*web server internal Yahoo*), Tengine (varian dari Nginx), dan IBM Servers (varian dari Apache). Dari kesepuluh aplikasi *web server* ini hanya lima yang berlisensi Open Source [11] sehingga bisa dengan mudah didapatkan, yaitu: Apache, Nginx, Tomcat, Lighttpd, dan Tengine. Tetapi karena Tomcat pada dasarnya adalah Apache dan Tengine maka penelitian ini hanya memfokuskan diri kepada aplikasi *web server*: Apache, Nginx, dan Lighttpd. Dalam penelitian ini, kinerja dari ketiga aplikasi *web server* ini diuji untuk melihat apakah layak untuk dijalankan pada Raspberry Pi yang dilanjutkan dengan membandingan kinerja masing-masing aplikasi *web server* untuk menentukan aplikasi yang mana yang paling cocok untuk dijalankan pada Raspberry Pi.

Salah satu topologi yang umum dipakai untuk perangkat *web server* adalah dengan menempatkan aplikasi *web server* dan halaman-halaman statis atau program yang akan membangkitkan halaman-halaman dinamis pada perangkat yang sama. Tetapi karena terbatasnya kapasitas penyimpanan sekunder yang dimiliki oleh Raspberry Pi topologi ini tidak mungkin dipakai. Topologi alternatif adalah dengan menempatkan halaman-halaman statis atau program yang akan membangkitkan halaman-halaman dinamis pada *hardware* terpisah dengan *hardware* yang menjalankan aplikasi *web server*-nya tetapi tetap dalam satu jaringan lokal yang sama. Topologi ini dapat dilihat pada Gambar 1. Dalam topologi alternatif ini aplikasi *web server* akan berjalan di Raspberry Pi tetapi semua halaman-halaman statis atau program yang akan membangkitkan halaman-halaman dinamis ditempatkan pada sebuah Network File System (NFS) yang sama-sama berada pada satu jaringan lokal. Penelitian ini memakai topologi alternatif ini dalam menguji kelayakan dan kinerja aplikasi *web server*.

Dua metrik utama dalam pemilihan suatu aplikasi *web server*, biasanya, adalah jumlah maksimum permintaan

simultan yang bisa dilayani dalam suatu waktu tertentu (*maximum request*) dan waktu yang dibutuhkan untuk melayani permintaan-permintaan yang diterimanya (*reply time*). Jika jumlah *maximum request* semakin besar maka suatu perangkat dapat melayani lebih banyak permintaan dalam kurun waktu tertentu. Sedangkan jika *reply time* semakin kecil maka waktu tunggu untuk mengirim balasan ke *web browser* akan semakin pendek dan ini penting untuk kepuasan pemakaian karena akan mengurangi waktu tunggu para pemakai dalam mendapatkan balasan dari aplikasi *web server*. Penelitian ini mengkaji kinerja *web server* dengan memakai kedua metrik ini.

## III. METODE

Seperti telah disebutkan di atas, penelitian ini memakai topologi yang terlihat pada Gambar 1. Spesifikasi *hardware* yang dipergunakan adalah: hub 1000 Mbps, satu PC biasa yang berfungsi sebagai *client*, satu PC biasa sebagai NFS, dan satu Raspberry Pi. Semua *hardware* ini saling terhubung dalam satu jaringan lokal dengan memakai kabel UTP CAT5E. Raspberry Pi menjalankan sistem operasi Raspbian, yang merupakan sistem operasi khusus dipaketkan untuk *hardware* ini [12]. Sedangkan kedua PC menjalankan sistem operasi Ubuntu Desktop versi 12.10.

***index.html***

```
<!doctype html>
<html lang="en">
    <head>
        <title>Hello world...</title>
         <link rel="stylesheet"
               href="css/uji.css">
    </head>
    <body onload="uji();">
         <h1>Hello world...</h1>
    <script src="js/uji.js"></script>
    </body>
</html>
```

***uji.js***

```
function uji()
{
  alert("Hello world...");
}
```

***uji.css***

```
h1
{
  color: blue;
}
```

Program 1. Kode halaman statis

***index.php***

```
<?php
  phpinfo();
?>
```

Program 2. Kode halaman dinamis

Untuk software penguji kinerja aplikasi *web server* penelitian ini menggunakan JMeter [13] yang dijalankan pada komputer *client*. Dalam proses pengujian, JMeter akan membangkitkan sejumlah permintaan simultan ke aplikasi *web server* untuk mensimulasikan sejumlah user yang secara bersamaan melakukan permintaan kepada aplikasi *web server* melalui *web browser* mereka. Kode halaman statis yang dipakai untuk semua pengujian dapat dilihat pada Program 1 sedangkan kode halaman dinamis yang dipakai dapat dilihat pada Program 2. Seperti terlihat pada Program 2, halaman-halaman dinamis dibangkitkan dengan memakai PHP yang juga dijalankan pada Raspberry Pi. PHP yang dipergunakan adalah PHP versi 5.3.2.

Untuk menguji kelayakan suatu aplikasi *web server* maka Raspberry Pi, secara bergantian, diinstalasi dengan setiap jenis aplikasi *web server* yang diuji, yaitu: Apache, Nginx, dan Lighttpd. Setiap instalasi aplikasi *web server* memakai konfigurasi standar bawaannya. Raspberry Pi kemudian dimasukkan ke dalam topologi seperti pada Gambar 1 dan JMeter dijalankan pada *client* untuk mengirimkan sejumlah permintaan ke aplikasi *web server*.

Aplikasi *web server* dikatakan layak untuk dijalankan pada Raspberry Pi jika bisa menangani *minimum request* tanpa kesalahan (*error*) dalam maksimum *reply time* tertentu. Karena standar *minimum request* yang harus dipenuhi oleh suatu aplikasi *web server* tidak ada, dalam penelitian ini nilai *minimum request* ini ditetapkan sebesar 100 permintaan simultan. Besaran nilai *minimum request* ini ditetapkan dengan pertimbangan sebagai berikut: suatu aplikasi *web server* (menurut kami) minimal harus bisa melayani 15 pemakai secara simultan dan karena rata-rata setiap *web browser* hanya diperbolehkan membuka secara bersamaan maksimum enam koneksi ke satu *hostname* [14] maka jumlah minimal permintaan simultan (i.e. request) yang semestinya bisa ditangani oleh suatu aplikasi *web server* adalah 90 (15 pemakai x 6 koneksi) yang kemudian dibulatkan menjadi 100 permintaan simultan. Demikian juga dengan *reply time*, karena tidak ada standar nilai maksimumnya maka, penelitian ini memakai maksimum *response time* untuk situs yang dianjurkan oleh bidang HCI, yaitu satu detik [15].

Untuk menentukan jenis aplikasi *web server* mana yang paling cocok dijalankan pada Raspberry Pi maka, secara bergantian, setiap jenis aplikasi *web server* diuji kinerjanya dengan mencari nilai *maximum request* yang bisa ditanganinya. Sedangkan waktu *reply time* diambil saat aplikasi *web server* menangani *maximum request* tersebut. Hasil ini kemudian dibandingkan untuk melihat aplikasi *web server* yang mana yang memiliki kinerja terbaik. Dalam pencarian ini juga, setiap aplikasi *web server* yang diuji memakai konfigurasi standar bawaannya.

Proses pencarian nilai *maximum request* dilakukan melalui pengujian oleh JMeter yang akan mengirimkan sejumlah permintaan secara simultan kepada aplikasi *web server* yang sedang diuji. Dari balasan aplikasi *web server*, atas permintan-permintaan ini, JMeter akan mendapatkan parameter-parameter yang dapat dianalisis, seperti: *throughput*, *reply time*, *median*, *min*, *max* dan

Tabel 1. Hasil pengujian kelayakan untuk menjalankan aplikasi pada web server

| Aplikasi Web Server | Request | Halaman Statis | | | Halaman Dinamis | | |
|---|---|---|---|---|---|---|---|
| | | Error Rate (%) | Reply Time (ms) | | Error Rate (%) | Reply Time (ms) | |
| | | | Rata-rata | SD | | Rata-rata | SD |
| Apache | 100 | 0.00 | 180,10 | 31,51 | 0,00 | 3131,57 | 1133,03 |
| Lighttpd | 100 | 0.00 | 52,87 | 9,13 | 0,00 | 1434,43 | 25,28 |
| Nginx | 100 | 0.00 | 94,97 | 24,48 | 0,00 | 1249,13 | 3601,53 |

*error-rate* dan lain-lain. Jumlah permintaan simultan yang dibangkitkan oleh JMeter pada penelitian ini dimulai dari 100 permintaan simultan dan meningkat dengan kelipatan 100 permintaan sampai ditemukannya nilai permintaan simultan yang akan menimbulkan kesalahan (*error-rate* > 0). Nilai permintaan simultan saat pertama sekali terjadi error inilah yang diambil sebagai nilai *maximum request* aplikasi *web server* tersebut. Untuk setiap nilai permintaan simultan yang diuji, proses pengujiannya akan diulangi sebanyak 30 kali.

## IV. Hasil dan Pembahasan

### A. Kelayakan Menjalankan Aplikasi Web server

Hasil pengujian untuk melihat apakah Raspberry Pi layak untuk menjalankan aplikasi *web server*, baik untuk halaman statis maupun halaman dinamis, dapat dilihat pada Tabel 1. Dari hasil pengujian ini tampak bahwa untuk halaman statis semua aplikasi *web server* yang diuji berhasil melayani 100 permintaan secara simultan tanpa kesalahan dan dengan *reply time* sangat layak karena kesemuanya mampu melayani dibawah 1 detik, tepatnya: 0,18 detik untuk Apache, 0,05 detik untuk Lighttpd, dan 0,09 detik untuk Nginx.

Sedangkan untuk halaman-halaman dinamis, yang dibangkitkan oleh PHP, ketiga aplikasi webserver juga berhasil melayani 100 permintaan secara simultan tanpa kesalahan. Khusus untuk Nginx dan Lighttpd *reply time* dalam melayani ke-100 permintaan ini bisa dikategorikan layak karena mendekati 1 detik, tepatnya: 1,25 detik untuk Nginx dan 1,43 detik untuk Lighttpd. Sedangkan untuk Apache membutuhkan waktu yang lebih lama dan melebihi 1 detik, tepatnya: 3,13 detik.

Hasil pada Tabel 1 menunjukkan bahwa Nginx dan Lighttpd layak untuk dijalankan pada Raspberry Pi baik untuk melayani halaman-halaman statis maupun halaman-halaman dinamis, walaupun *reply time*-nya sedikit melebih 1 detik saat menangani halaman-halaman dinamis. Sedangkan untuk Apache di Raspberry Pi lebih layak dipakai untuk menyajikan halaman-halaman statis dan kurang layak untuk menyajikan halaman-halaman dinamis karena memberikan *reply time* yang relatif lama, tepatnya 3.13 detik.

### B. Perbandingan Kinerja Aplikasi Web server

Tabel 2 menunjukan jumlah permintaan simultan yang pertama sekali menimbulkan kesalahan untuk masing-masing aplikasi *web server* atau dengan kata lain ini merupakan batas atas permintaan simultan yang bisa dilayani oleh tiap-tiap aplikasi *web server*; yang juga disebut dengan istilah *maximum request*. Untuk halaman-halaman statis Nginx menduduki urutan pertama dengan *maximum request* sebesar 5.300 dan dapat melayani permintaan-permintaan tersebut (i.e. *reply time*) rata-rata dalam 1,26 detik. Sedangkan Lighttpd menduduki urutan kedua dengan *maximum request* sebesar 3.000 tetapi dengan *reply time* rata-rata yang sangat cepat yaitu 0,18 detik. Dan urutan terakhir diduduki oleh Apache dengan *maximum request* terkecil yaitu 2.900 dan *reply time* rata-rata yang relatif besar yaitu 2,85 detik.

Untuk halaman-halaman dinamis Nginx tetap memberikan *maximum request* terbesar yaitu 3.000 dengan *reply time* 1,07 detik. Apache menduduki urutan kedua dengan *maximum request* sebesar 400 tetapi dengan *reply time* yang sangat besar yaitu 8,70 detik. Sedangkan Lighttpd menduduki urutan terakhir dengan *maximum request* terkecil yaitu sebesar 200 dan dengan *reply time* yang relatif lambat yaitu 2,24 detik.

Hasil pada Tabel 2 menunjukkan bahwa Nginx merupakan aplikasi *web server* yang memiliki kinerja terbaik untuk dijalankan pada Raspberry Pi, baik untuk melayani halaman-halaman statis maupun untuk melayani halaman-halaman dinamis. Disamping mampu melayani permintaan simultan terbanyak, 5.300 untuk halaman statis dan 3.000 untuk halaman dinamis, *reply time* rata-rata Nginx dalam melayani kesemua permintaan tersebut bisa dikatakan layak karena masih medekati 1 detik, yaitu: 1,26 detik untuk halaman-halaman statis dan 1,07 detik untuk halaman-halaman dinamis.

Menarik untuk diperhatikan nilai *maximum request* untuk halaman-halaman dinamis 56,60% lebih kecil dari *maximum request* untuk halaman-halaman statis, 3.000 dibanding 5.300. Ini diperkirakan karena untuk halaman-

Tabel 2, Hasil pencarian maximum request dan reply time aplikasi web server

| Aplikasi Web Server | Halaman Statis | | | | Halaman Dinamis | | | |
|---|---|---|---|---|---|---|---|---|
| | Maximum Request | Error Rate (%) | Reply Time (ms) | | Maximum Request | Error Rate (%) | Reply Time (ms) | |
| | | | Rata-rata | SD | | | Rata-rata | SD |
| Apache | 2900 | 0,01 | 2852,30 | 1313,44 | 400 | 0,01 | 8695,90 | 2448,11 |
| Lighttpd | 3000 | 0,01 | 180,50 | 5,59 | 200 | 0,07 | 2242,53 | 3547,26 |
| Nginx | 5300 | 0,01 | 1256,27 | 83,52 | 3000 | 0,01 | 1074,63 | 583,09 |

halaman dinamis Raspberry Pi harus menjalankan dua aplikasi sekaligus, yaitu Nginx dan PHP, sehingga akan mengurangi kinerja dari prosesornya yang dalam hal ini hampir setengahnya.

Buruknya kinerja Apache dan Lightpd dibandingkan Nginx pada Raspbery Pi diperkirakan karena Apache dan Lighttpd dirancang untuk dijalankan pada *hardware* yang bisa melakukan *multithread* [10] sedangkan Nginx dirancang untuk dijalankan sebagai aplikasi *single thread* [16]. Dalam hal ini *hardware* Raspberry Pi tidak mendukung *multithread* [17] tetapi *multithread* di Raspberry Pi dilakukan secara software oleh sistem operasinya yang mengakibatkan kinerja *multithread*-nya lebih lambat dibanding *multithread* yang langsung dilakukan oleh *hardware*. Dengan demikian saat Apache dan Lighttpd dijalankan pada Raspberry Pi ada proses-proses tambahan yang perlu dilakukan oleh sistem operasi untuk mengakomodir kebutuhan *multithread* aplikasi-aplikasi ini. Sedangkan proses-proses tambahan ini tidak ada saat Nginx dijalankan, karena bersifat *single thread*, sehingga dapat dimanfaatkan oleh Nginx untuk melayani lebih banyak permintaan yang datang kepadanya.

## V. KESIMPULAN

Dari hasil pengujian yang dilakukan, Raspberry Pi layak dijadikan *web server* terutama untuk aplikasi-aplikasi *web server* berikut: Apache, Lighttpd, dan Nginx. Ketiga aplikasi *web server* ini dapat menangani 100 permintaan simultan, baik untuk halaman-halaman statis maupun halaman-halaman dinamis, dalam kurun waktu 1 detik. Tetapi diantara ketiga aplikasi *web server* ini, Nginx-lah yang memiliki kinerja terbaik di Raspberry Pi karena mampu menangani 5.300 permintaan simultan untuk halaman-halaman statis dan 3.000 permintaan simultan untuk halaman-halaman dinamis serta kesemua permintaan ini dapat dilayani dalam kurung waktu rata-rata mendekati 1 detik.

Halaman-halaman dinamis dalam pengujian yang dilakukan dibangkitkan oleh aplikasi PHP. Tetapi PHP belum tentu aplikasi pembangkit halaman-halaman dinamis yang paling tepat untuk dijalankan pada Raspberry Pi sehingga perlu dilakukan penelitian lanjutan untuk membandingkannya dengan aplikasi sejenis lainnya, seperti: Node.js, Ruby on Rail, Flask, dan Tomcat. Disamping itu Nginx juga dimungkinkan untuk di-*cluster*-kan sehingga menarik untuk melihat peningkatan kinerja Nginx sejalan dengan jumlah Raspberry Pi dalam *cluster* tersebut.

## REFERENSI


[1] Dell, *PowerEdge R620 Technical Guide,* Dell, 2013.

[2] Dell, *Dell PowerEdge R620 Getting Started Guide*, Dell, 2012.

[3] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless Sensor Network Survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[4] K. Martinez, R. Ong, and J. Hart, "Glacsweb: a sensor network for hostile environments," in *the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004, pp. 81–87.

[5] K. Mase, "How to deliver your message from/to a disaster area," *IEEE Commun. Mag.*, vol. 49, no. 1, pp. 52–57, Jan. 2011.

[6] M. Richardson and S. Wallace, *Getting Started with Raspberry Pi*, Sebastopol, CA: O'Reilly Media, 2013.

[7] E. Upton and G. Halfacree, *Raspberry Pi User Guide*, West Sussex, England: Wiley, 2012.

[8] W3Techs. (view: 17 July 2014). Web Technology Market Report: Usage of *web server*s for websites [Online]. Available: http://w3techs.com/technologies/overview/web_server/all.

[9] Open Source *Hardware* Association, "Open-Source *Hardware*: Definition," Open Source *Hardware* Association, 17-Jul-2014. [Online]. Available: http://www.oshwa.org/definition/.

[10] B. Laurie and P. Laurie, *Apache: The Definitive Guide*, O'Reilly Media, 2002.

[11] Open Source Initiatives (view: 17 July 2014). The Open Source Definition, Open Source Initiatives [Online]. Available: http://opensource.org/definition.

[12] Raspbian Development Team, Raspbian.

[13] The Apache Software Foundation, Apache JMeter. The Apache Software Foundation.

[14] Browserscope (view: 17 July 2014). Browserscope: Network: Connection per Hostname [Online]. Available: http://www.browserscope.org.

[15] J. Nielsen (view: 17 July 2014). Website *Response time*s [Online]. Available: http://www.browserscope.org.

[16] C. Nedelcu, *Nginx HTTP Server*, Birmingham, UK: Packt Publishing Ltd., 2013.

[17] marwan90, "Does Raspberry Support *Multithread*ing?," Raspberry Pi Forum, 2012.