PERANCANGAN DAN IMPLEMENTASI PENGGUNAAN CLOUD SERVER PAAS (PLATFORM AS A SERVICE) SEBAGAI PENYIMPAN DATA PENGUNJUNG DAN REKAMAN VIDEO PENGUNJUNG PADA BUILDING SECURITY SYSTEM BERBASIS EMBEDDED

DESIGN AND IMPLEMENTATION CLOUD SERVER PAAS (PLATFORM AS A SERVICE) AS VISITOR'S DATABASE AND VISITOR'S VIDEO DATABASE ON BUILDING SECURITY SYSTEM BASED ON EMBEDDED

Robby Reza[1], Agung Nugroho Jati, ST., MT[2], Umar Ali Ahmad, ST., MT[3]

[1,3]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]robbyreza@students.telkomuniversity.ac.id, [2]agungnj@telkomuniversity.ac.id, [3]uaa@ypt.or.id

Abstrak

Untuk menunjang sistem keamanan dibutuhkan *server* yang berfungsi sebagai penyimpan yang berupa rekaman *IP Camera* dan juga menyimpan *Record* data pengunjung. *Server* yang dipilih adalah *Cloud Server PaaS (Platform as a Service). Cloud Server* ini dibuat dengan menjadikan *Raspberry Pi* sebagai *Platform*. Pada *cloud server* akan dirancang *API* yang digunakan untuk mempermudah komunikasi antar *device*. *API* yang digunakan adalah *REST API* yang dibangun dengan menggunakan *framework Flask*.*Cloud Server* yang dibuat dapat menyimpan dan mengirim data yang diterima dari sistem *embedded* dan *IP Camera*. *Cloud Server* yang dibuat dapat diakses oleh user melalui aplikasi *mobile*. Saat melakukan akses ke layanan *server*, *user* akan mendapatkan kecepatan respon yang berbeda-beda. Perbedaan ini disebabkan oleh penggunaan layanan internet yang berbeda dalam hal ini wired dan wireless. Dari hasil penelitian, waktu respon rata-rata menggunakan fiber optik adalah 624.8667 ms dan respon rata-rata menggunakan HSPA adalah 2224.333 ms. Dan *Cloud Server* ini hanya dapat melayani pengguna yang terbatas dalam satu waktu yaitu maksimal 147 *request* dalam satu waktu.

Kata Kunci: *cloud server, building security system*

Abstract

To support this security system need a server that serves as a storage form of recording and storing IP Camera Record visitor data. The selected server is Cloud Server PaaS (Platform as a Service). Cloud Server is made to make the Raspberry Pi as a Platform. In the cloud server will be designed API that is used to facilitate communication between devices. API that used is a REST API that built using Flask framework. This facilitates the use of REST APIs to perform data communications used for communication using the commands in HTML. Cloud servers are made accessible by the user via the mobile application. When performing access to the server, the user will get a response speed is different. This difference is caused by the use of different internet service in this wired and wireless. From the research results by using the fiber optic services faster; the average 624.8667 ms response time, compared to the use of HSPA has the average response time 2224.333 ms. And Cloud Server can only serve a limited user at a time is a maximum of 147 requests at one time.

Keyword : *cloud server, building security system*

1. Pendahuluan

Berbagai jenis teknologi dengan kecanggihan yang mutakhir telah banyak diciptakan oleh manusia untuk mempermudah segala kegiatan yang dilakukannya. Salah satu teknologi yang berkembang yaitu teknologi di bidang pengamanan. Teknologi ini berkembang tahap demi tahap dari yang sederhana berupa *alarm* yang menandakan jika ada seseorang yang mecoba melewati atau mencoba masuk ke suatu ruangan tertentu hingga sistem keamanan yang canggih misalnya yang terintegrasi dengan sensor infra merah dan kamera *CCTV* .Salah satu sistem keamanan yang dibuat adalah sistem keamanan berbasis embedded. Sistem keamanan ini menggunakan mikroprosesor sebagai pengolah informasi.

Untuk menunjang sistem keamanan ini dibutuhkan *server* yang berfungsi sebagai penyimpan yang berupa rekaman *IP Camera* dan juga menyimpan *Record* Data pengunjung. *Server* yang dipilih adalah *Cloud Server PaaS*

(Platform as a Service). Cloud Server ini dibuat dengan menjadikan *Raspberry Pi* sebagai *Platform*. Pada cloud server akan dirancang *API* yang digunakan untuk mempermudah komunikasi antar *device*. *API* yang digunakan adalah *REST API* yang dibangun dengan menggunakan *framework Flask*. Penggunaan *REST API* ini memudahkan dalam melakukan komunikasi data karena komunikasi yang digunakan menggunakan perintah-perintah yang ada di *HTML*. Dengan digunakannya *Cloud Server* ini dapat mempermudah dalam mengakses data dari mana saja.

2. Tinjauan Pustaka

2.1 Cloud Computing

Cloud Computing adalah suatu bentuk komputasi yang memungkinkan dapat diakses dari manapun yang memberikan kenyamanan dan akses jaringan sesuai permintaan ke lokasi sumber daya komputasi yang telah dikonfigurasi (misalnya jaringan, *server*, penyimpanan, aplikasi dan layanan) yang dapat digunakan dengan cepat dan sedikit interaksi dengan penyedia layanan.[4]

2.1.1 Layanan *Cloud Computing*

Ada tiga layanan dasar yang ditawarkan dalam komputasi awan, yaitu *Software As A Service, Platform As A Servi.ce* dan *Infrastructure As A Service*.

1. IaaS (Infrastructure As A Service)

Pada *IaaS*, penyedia komputasi awan menyediakan infrastruktur seperti sumber daya komputasi, media penyimpanan, dan jaringan kepada pengguna, seolah-olah penyedia menyediakan satu komputer *virtual* kosong yang dapat diisi sesuai keinginan pengguna yang dapat digunakan oleh pengguna melalui jaringan secara *virtual*. Penyedia layanan juga dapat menawarkan sistem operasi yang dapat dipasang pada komputer *virtual* tersebut. [4]

2. PaaS (Platform As A Service)

Pada *PaaS* penyedia jasa akan memberikan spesifikasi perangkat keras yang sesuai kebutuhan aplikasi yang akan dikembangkan pengguna. Ini seperti *IaaS* yang telah dipasang sistem operasi dan pengguna tinggal memilih perangkat lunak untuk mengembangkan aplikasi. Pengembanganan membuat aplikasi pada *platform* penyedia yang dilakukan melalui jaringan internet. Namun, pengguna tidak dapat mengatur sumber daya komputasi seperti *memory*, media penyimpanan, processing power.[4]

3. *SaaS* (*Software As A Service*)

Pada *SaaS* penyedia memiliki kemampuan mendistribusikan aplikasi kepada pengguna. Aplikasi dapat diakses oleh pengguna melalui aplikasi *web browser* yang ada pada komputer pengguna. Pengguna tidak dapat mengatur dan tidak memiliki kontrol terhadap sistem operasi, media penyimpanan, *server*, maupun jaringan.[4]

2.2 Flask

Ada beberapa framework yang menggunakan bahasa *python* sebagai basisnya, diantaranya Django, Flask, Pyramid, Tornado, Bottle, Diesel, Pecan, Falcon dan yang lainnya.Pada tulisan ini, penulis ingin membahas tentang penggunaan Flask sebagai *framework* untuk menunjang *cloud server* yang dibuat. Flask adalah *microframework* berbasis *python* yang dipelopori oleh Armin Ronacher. Bila dibandingkan dengan Django, Flask jauh lebih ringan dan cepat karena Flask dibuat dengan ide menyederhanakan inti *framework*-nya seminimal mungkin. Dengan *tagline "web development, one drop at a time"*, *Flask* dapat membantu kita membuat situs dengan sangat cepat meskipun dengan *library* yang sederhana.[5]

2.3 REST

Representational State Transfer atau biasa disebut *REST* adalah suatu arsitektur metode komunikasi yang sering diterapkan dalam pengembangan layanan berbasis *web* arsitektur. *REST* yang umumnya dijalankan *via HTTP* (*Hypertext Transfer Protocol*), melibatkan proses pembacaan laman web tertentu yang memuat sebuah *file XML* atau *JSON*. File inilah yang menguraikan dan memuat konten yang hendak disajikan. Setelah melalui sebuah proses definisi tertentu, user akan bisa mengakses antarmuka aplikasi yang dimaksudkan. kekhasan REST terletak pada interaksi antara klien dan server yang difasilitasi oleh sejumlah tipe operasional (verba) dan *Universal Resource*

Identifiers (*URIs*) yang unik bagi tiap-tiap sumberdaya. Masing-masing verba – *GET*, *POST*, *PUT* dan *DELETE* – memiliki makna operasional khusus untuk menghindari ambiguitas.[9]

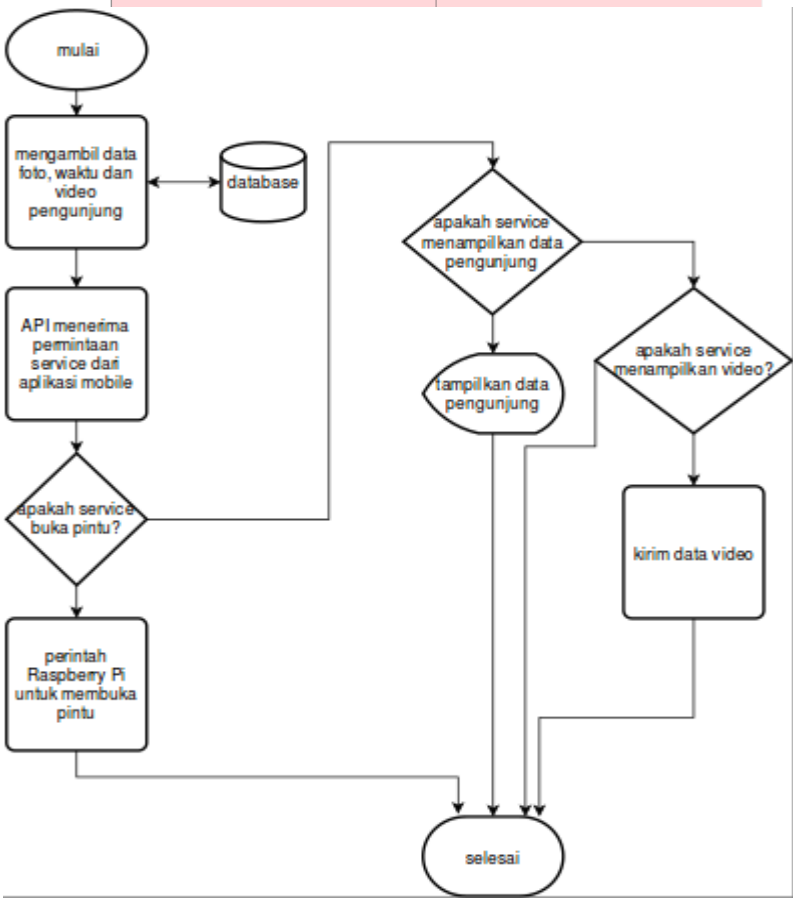
REST adalah gaya arsitektural yang memiliki aturan seperti antar muka yang seragam, sehingga jika aturan tersebut diterapkan pada *web services* akan dapat memaksimalkan kinerja *web services* terutama pada performa, skalabilitas, dan kemudahan untk dimodifikasi. Pada arsitektur *REST* data dan fungsi dianggap sebagai sumber daya yang dapat diakses lewat *Uniform Resource Identifier* (*URI*), biasanya berupa tautan pada *web*.

Dalam penerapannya, *REST* lebih banyak digunakan untuk *web service* yang berorientasi pada *resource*. Maksud orientasi pada sumber data adalah orientasi yang menyediakan sumber data sebagai layanannya dan bukan kumpulan – kumpulan dari aktifitas yang mengolah sumber daya itu.Bentuk *web servise* menggunakan *REST style* sangat cocok digunakan sebagai *backend* dari aplikasi berbasis mobile karena cara aksesnya yang mudah dan hasil data yang dikirimkan berformat *JSON* sehingga ukuran *file* menjadi lebih kecil.[9]

2.4 Perancangan

Diagram Alir perancangan sistem

Berikut adalah diagram alir pada proses akses service pada *cloud server*



Gambar 2.4 diagram alir akses *service* buka pintu pada *cloud server*

Data dari sistem akan disimpan pada database yang telah dibuat agar dapat diakses melalui API yang telah dibuat. API kemudian menerima permintaan *service* dari aplikasi *mobile*. API akan menilai apakah *service* yang diminta merupakan permintaan membuka pintu. Jika ya, maka API akan memberikan perintah untuk membuka pintu.

Apabila *service* yang diminta merupakan perminataan menampilkan data pengunjung maka API akan memberikan menampilkan data pengunjung pada halaman *web*. Apabila permintaan *service* merupakan permintaan menampilkan video, maka API akan mengirim data video pada aplikasi mobile.

3. Pembahasan

Layanan yang dibuat merupakan mengadopsi arsitektur *REST API*. Layanan ini dibuat dengan menggunakan *framework flask* yang menggunakan bahasa pemrograman *python*. *User* nantinya dapat memberikan perintah pada *Raspberry Pi* melalui layanan ini. Perintah nantinya dikirim dengan *HTTP POST* melalui *URI* yang telah disediakan. *Software tunneling* digunakan agar API yang dibuat dapat diakses melalui internet. Salah satu contoh perintah pada *Raspberry Pi* yaitu membuka pintu. Pengujian ini dilakukan dengan cara memberikan *request* sebanyak banyaknya pada *server* dan melihat seberapa banyak data yang dapat di *replies* kembali

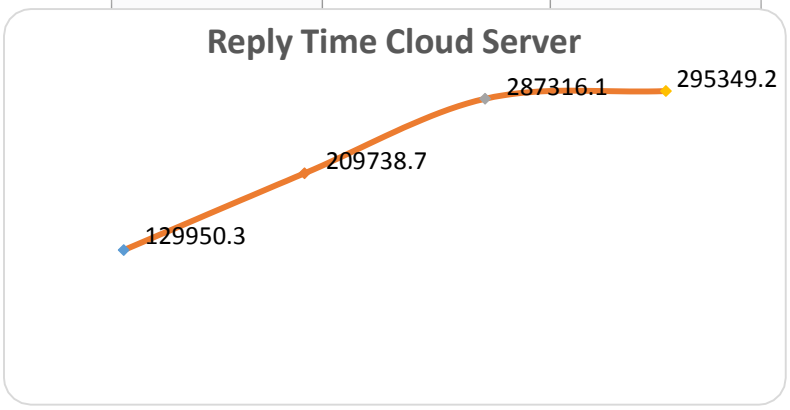
3.1 Pengujian Beban Server

Pengujian ini dilakukan dengan cara memberikan *request* sebanyak banyaknya pada *server* dan melihat seberapa banyak data yang dapat di *replies* kembali. Pengujian ini bertujuan untuk mengetahui seberapa banyak permintaan yang dapat dilayani oleh *server* dalam satu waktu. Pengujian akan dilakukan dengan aplikasi *htterf* yang merupakan *software* untuk menguji beban yang dapat ditampung *server*. Pengujian dilakukan pada *URI* /pengunjung yang merupakan *URI* untuk data pengunjung. Pada pengujian ini dilakukan pengiriman *request* sebanyak 50, 100, 150 dan 200 *request* untuk mengetahui seberapa banyak *replies* yang berikan oleh *server*. Berikut adalah hasil pengujian yang didapatkan

Tabel 3.1 Hasil pengujian beban server menggunakan httperf

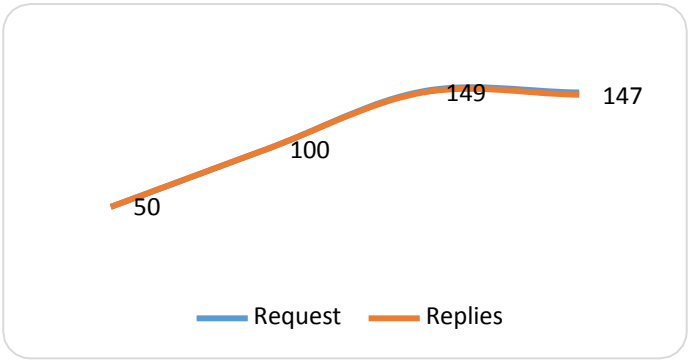
Connections	Request	Replies	Transfer Time (ms)	Reply Time (ms)
50	50	50	4288.9	129950.3
100	100	100	3264.4	209738.7
150	150	149	2910.1	287316.1
200	149	147	3083.4	295349.2

Pada pengujian pengiriman *request*, dari 50 koneksi yang dibangun didapatkan hasil bahwa dari 50 *request*, 50 *request* yang dapat dilayani dan memiliki *transfer time* 4288.9 ms serta nilai *reply time* sebesar 129950.3 ms. Pada pengujian kedua, dari 100 koneksi yang dibangun didapatkan hasil bahwa dari 100 *request*, 100 *request* yang dapat dilayani dan memiliki transfer time 3264.4 ms serta nilai *reply time* sebesar 209738.7 ms. Pada pengujian ketiga, dari 150 koneksi yang dibangun didapatkan hasil bahwa dari 150 *request*, 149 *request* yang dapat dilayani dan memiliki transfer time 2910.1 serta nilai *reply time* sebesar 287316.1 ms. Pada pengujian keempat, dari 200 koneksi yang dibangun didapatkan hasil bahwa hanya 149 *request* yang dapat dilakukan, 147 *request* yang dapat dilayani dan memiliki *transfer time* 3083.4 ms serta nilai *reply time* sebesar 295349.2 ms. Dari hasil tersebut didapatkan grafik sebagai berikut



Gambar 3.1 Grafik Reply Time Cloud Server

Pada grafik diatas dapat terlihat perbedaan yang signifikan antara transfer time dan *reply time*. Perbedaan ini diakibatkan karena *API* bersifat *synchronous* sehingga penanganan permintaan akan dilakukan setelah permintaan sebelumnya diselesaikan. Semakin banyak request yang terjadi pada *server* dalam satu waktu, maka semakin besar pula waktu yang dibutuhkan untuk melakukan balasan pada pengguna.



Gambar 3.2 Kemampuan Melayani Request pada Cloud Server

Pada grafik diatas dapat dilihat, dari 50 *request* hingga 100 *request cloud server* dapat melayan permintaan dengan baik. Tetapi terjadi penurunan kinerja *server* pada saat request mencapai 150 *request*. Penurunan ini terjadi karena keterbatasan *processor* yang dimiliki *Raspberry Pi* dalam menyajikan layanan terhadap beberapa permintaan sekaligus. Tetapi untuk penggunaan yang terbatas, *Raspberry Pi* masih dapat dijadikan alternatif *platform* untuk layanan *cloud server*.

4. Kesimpulan dan Saran

4.1 Kesimpulan

Cloud Server dengan menggunakan *Raspberry Pi* sebagai *Platform* hanya dapat melayani dibawah 200 request. Keterbatasan ini disebabkan kecilnya *clock speed* pada *processor Raspberry Pi* yaitu 700 Mhz

4.2 Saran

Menambahkan sistem *failover* yang dapat berfungsi apabila server mengalami kegagalan menyediakan layanan.

DAFTAR PUSTAKA

[1] Bhuvaneswari.S, Sahaya Anselin Nisha.A,2013**.** *Implementation of Tcp/Ip on Embedded Webserver Using aspberry Pi In Industrial Application***.** India:Sathyabama University, Chennai, India
[2] Ibrahim, R. ; Zin, Z.M. *Study of automated face recognition system for office door access control application IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, 2011.
[3] Md. Nasimuzzaman Chowdhury, Md. Shiblee Nooman, Srijon Sarker, 2013. *Access Control of Door and Home Security by Raspberry Pi Through Internet*. Bangladesh: AIUB (American International University-Bangladesh)
[4] Pratama, I Putu Agus Eka. 2014. *Smart City Beserta Cloud Computing dan Teknologi-Teknologi Pendukung Lainnya*.Bandung:Informatika.
[5] Brown, Ryan. 2015. *Django vs Flask vs Pyramid : Choosing a Python Web Framework*. [Online] Available at: https://www.airpair.com/python/posts/django-flask-pyramid [Accessed 12 December 2015].
[6] Rahman, Muhammad Aminudin. 2013. Perancangan dan Implementasi RESTful Web Service untuk Game Sosial Food Merchant Saga pada Perangkat Android. Teknik Informatika ITS.1:2
[7] Surendra, Martinus Raditia Sigit. 2014. Implementasi PHP Web Service Sebagai Penyedia Data Aplikasi Mobile. Sistem Informasi, Universitas Multimedia Nusantara.1:3
[8] Fielding,Roy Thomas.2000.Architectural Styles and the Design of Network-based Software Architectures. Disertasi Doktor pada UNIVERSITY OF CALIFORNIA, IRVINE: Tidak diterbitkan.
[9] Fielding, R. T.; Taylor, R. N. (2000). "Principled design of the modern Web architecture": 407–416.