UNIVERSITAS INDONESIA

ANALISIS PERENCANAAN KEBUTUHAN SISTEM *REAL COUNT* KOMISI PEMILIHAN UMUM (KPU)

TRANG WISESA WARDHANI 1606965505
EKO ADITYA BAGUS S 1606964982
HAFIDH FINANDRIYANTO 1606965045

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI MAGISTER TEKNOLOGI INFORMASI
JAKARTA
APRIL 2017

Problem Overview

Sebagai perwujudan transparansi dalam pemilu, Komisi Pemilihan Umum (KPU) membuat sistem untuk menampilkan rekapitulasi real count dari TPS-TPS yang diinput dari beberapa titik. Saat ini kendala yang dihadapi oleh KPU adalah sulitnya mengakses *server* pada hari pertama pemilu. Dikarenakan banyaknya antusiasme masyarakat untuk melihat hasil pemilu secara serempak pada waktu yang berdekatan menyebabkan permintaan yang harus dilayani *server* melebihi angka yang mampu dilayani oleh *server,* sehingga *server* sulit untuk di akses. Hal juga mempersulit tim KPU ketika akan melakukan *input* rekapitulasi suara.

Problem Statement

KPU ingin sistem *real count* dapat beroperasi selama 7x24 jam tanpa ada gangguan akses seperti pada tahap pertama, sehingga masyarakat dapat mengetahui secara *real time* hasil dan status ter-update dari proses pemilu.

Pada saat ini atau tahap pertama, *server* sangat sulit diakses oleh masyarakat, padahal dari laporan utilisasi *bandwidth* masih belum digunakan secara maksimal. *Server* untuk aplikasi *real count* ini menggunakan 2 buah *server* fisik yang divirtualisasi, masing-masing untuk *server database* dan *server* aplikasi. Tiap-tiap *server* mendapat alokasi sebesar 4GB. Apabila KPU mengabaikan masalah ini, akan banyak kritik dari masyarakat yang mengakibatkan reputasi dari lembaga negara tersebut menjadi buruk.

List Requirement

Dari permasalahan tersebut, berikut ini adalah beberapa fitur yang harus dimiliki oleh sistem *real count* KPU :

1. *Operational Requirement*
    - Sistem *real count* KPU harus dapat terintegrasi dengan sistem informasi lainnya.
    - Sistem *real count* dapat diakses oleh user seluruh indonesia.
    - Sistem *real count* KPU harus dapat di akses melalui berbagai jenis perangkat seperti komputer, laptop, HP, tablet PC, dsb.

- Sistem *real count* KPU dapat di akses diberbagai browser
- Sistem KPU dapat beroperasi 24x7
- Sistem dapat melakukan *update* dan *maintenance* tanpa menggangu operasional *real count*

2. *Performance Requirement*
    - Waktu respon sistem aplikasi harus dapat menampilkan halaman *real count* kurang dari 3 detik pada saat jam sibuk.
    - Database real cout harus terupdate secara *real time*
    - Sistem harus memiliki 99 % *up time*
    - Sistem dapat menyimpan data minimal 127GB
    - Sistem *real count* dapat melakukan transaksi bersamaan sebanyak >100 user per detik

3 *Security Requirement*
    - Hanya pihak terkait yang dapat melakukan *input real count*
    - Pengguna umum hanya dapat melihat status *real count*
    - Pengiriman data antar TPS harus terenkripsi
    - File yang di *upload* harus secara otomatis dilakukan cek virus sebelum tersimpan ke sistem
    - Server real count dapat terhindar dari berbagai serangan *cyber* seperti *web deface* dan DDoS.

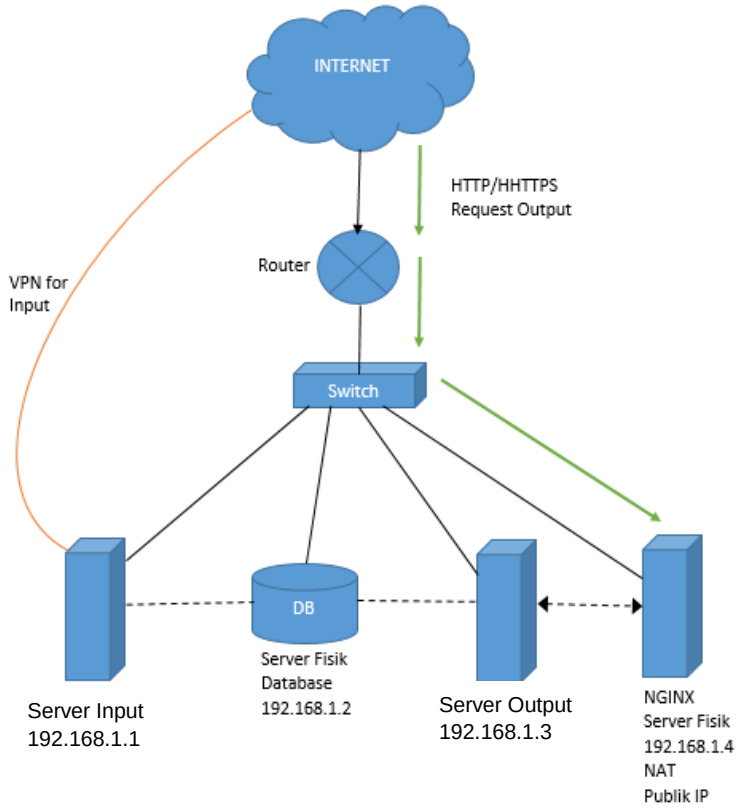
4 *Cultural and Political Requirement*
    - Aplikasi *real count* harus menggunakan bahasa Indonesia
    - Tidak boleh menyinggung agama atau etnis tertentu

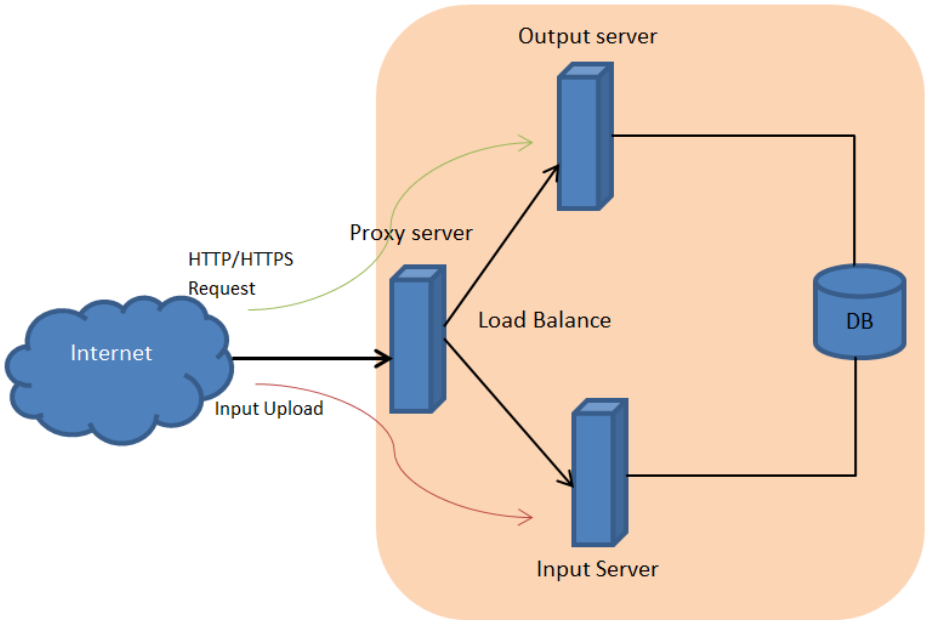
Saran

Berikut beberapa saran yang bisa dijadikan masukan KPU untuk merancang infrastruktur real count :

1. Memisahkan proses aplikasi *output real count* dengan aplikasi input data pada server Aplikasi.
2. Menggunakan solusi *reverse proxy*, sebagai *cache* dan *load balancing* untuk membagi *load server*, contoh NGINX , Varnish , LightHTTPD , HAProxy, Litespeed.
   misal *Apache* sebagai *backend* dan NGINX sebagai *frontend*.
3. Menggunakan solusi Load balanced Clustering untuk HA (High Availability) atau *Zero Downtime*.
   Misal , menggunakan NGINX dan Varnish sebagai Dual Proxy load balance.
4. Menggunakan solusi *Cloud* untuk *clustering* sebagai *backup*, misalnya menggunakan Amazon Elastic Load Balance.
5. Merubah konten halaman *real count* agar *size* yang di *load* berkurang.
6. Menggunakan *storage* khusus NAS, contoh TrueNAS
7. Melakukan load balance SQL database, contoh : *replication, database mirroring*
8. Menambahkan RAM pada server Aplikasi menjadi 8GB,
   Server fisik aplikasi saat ini menggunakan RAM 4GB , apabila menggunakan OS (30%) serta menjalankan virtualisasi maka :
   RAM yang tersisa = 4 x 30% = 2,8GB RAM.
   2,8GB RAM dibagi 2 untuk aplikasi Output dan Input masing2 = 1,4GB,
   Kebutuhan VM server aplikasi = 1,4GB x 30% = 1GB pembulatan.
   Jadi kesimpulannya , alokasi *memory* yang efektif digunakan masing-masing di VM aplikasi = 1GB .
   *Asumsi terdapat 100 *concurrent* user , jika rata-rata *memory* yang digunakan oleh halaman web sebesar 75MB, maka 100x75MB = 7,5GB *memory* (tidak termasuk OS)

➢ **Solusi Menggunakan Reverse Proxy**

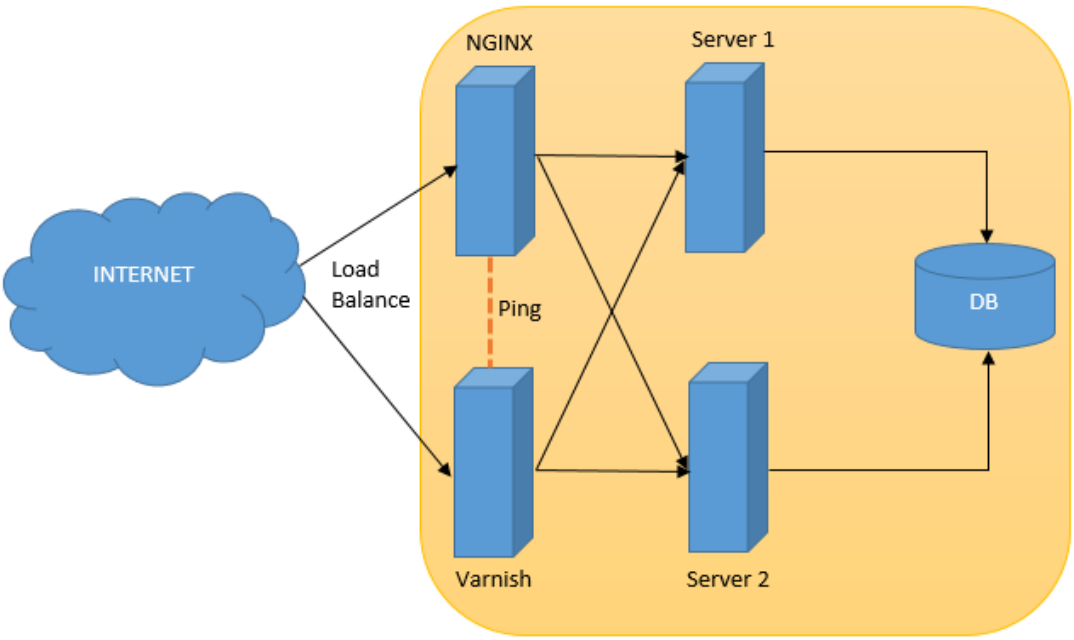




Keterangan:

Apabila menggunakan *proxy* maka *request* output HTTP/HTTPS akan diload balance oleh *server proxy*, dapat juga difungsikan untuk menampilkan konten halaman *web* (*cache*), Dengan demikian koneksi untuk input dan output dapat di bagi dan tidak diolah oleh 1 server fisik yang sama.

➢ **Solusi Menggunakan Load Balanced Clustering**



Keterangan:

Apabila menggunakan *Load Balanced Clustering* dengan *dual proxy (query round robin* atau dengan *priority)*, apabila salah satu *server proxy* mengalami *failure* , maka *server proxy* lainya akan *take over load balance session* , dengan demikian user tidak akan merasa mengalami *down time* sehingga *session* user tetap terjaga.