Server Load Balancer

國立陽明交通大學資工系資訊中心

Computer Center of Department of Computer Science, NYCU

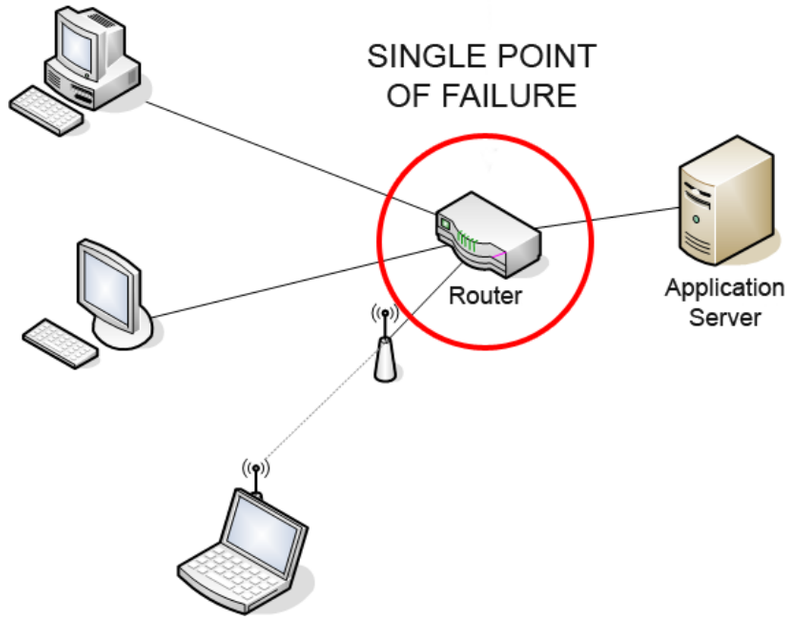
Introduction

- More users, more resources needed
  - CPU, RAM, HDD …
- Scale Up & Scale Out
  - One powerful server to service more users; or
  - Multiple servers to service more users
- Pros & Cons ?
- C10K Problem

Introduction

- High Availability
  - A characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.
- Availability (per year)
  - 99%: 3.65days
  - 99.9%: 8.77 hours (3 nines)
  - 99.99%: 52.60 minutes (4 nines)
  - 99.999%: 5.26 minutes (5 nines)

High Availability



Single point of failure - Wikipedia

- Principles
  - Elimination of single points of failure.
  - Reliable crossover.
    - Reliable configuration / topology change
  - Detection of failures as they occur.
- Graceful Degradation
  - the ability of a computer, machine, electronic system or network to maintain limited functionality even when a large portion of it has been destroyed or rendered inoperative.

Load Balancing

- Client Side
  - e.g: DNS round-robin
  - Pros & Cons
- Server Side
  - Server Load Balancer

Server Load Balancer (1)

- Provide “Scale-Out” and HA features
- Share loading among all backend nodes with some algorithms
  - Static Algorithms: does not take into account the state of the system for the distribution of tasks.
  - Dynamic Algorithms

Server Load Balancer (2)

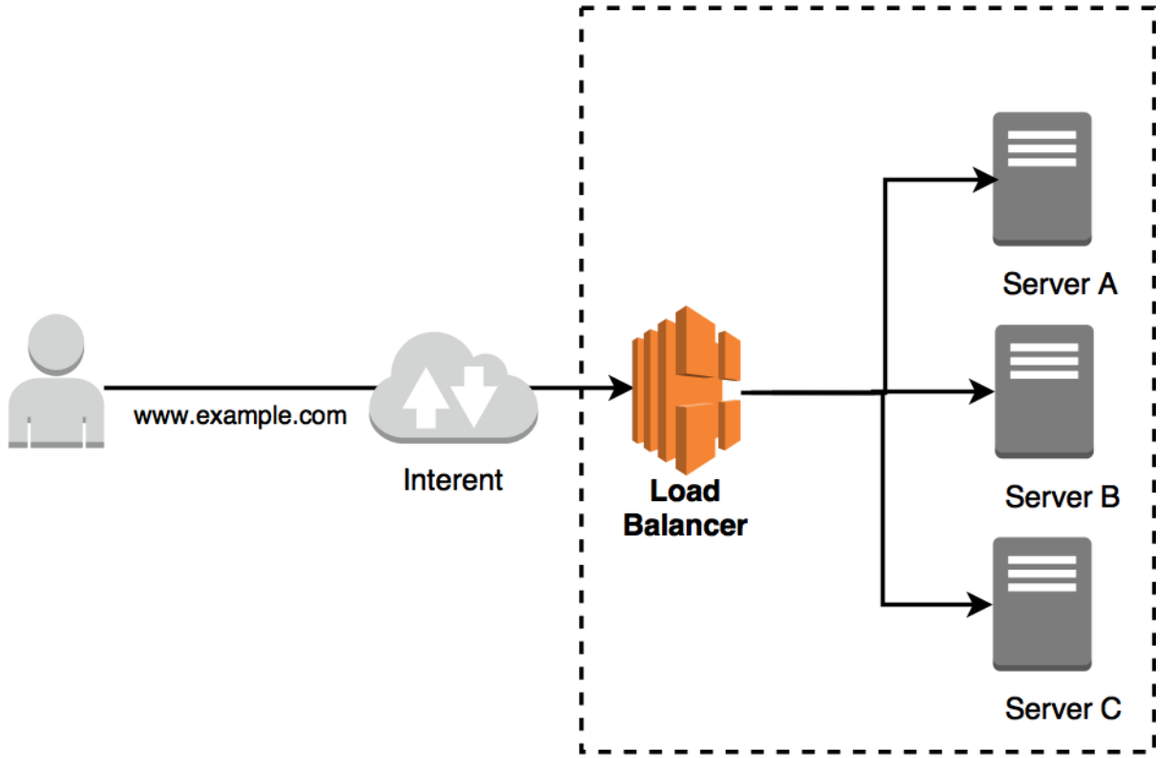
- Layer 4 or Layer 7
  - Layer 4 Switch
- Distribution Algorithms
  - Round-robin
  - Random
  - Ratio
  - Hash Table
  - Least-connections
  - Persistence
    - Session-ID (e.g. HTTP Cookie)

Server Load Balancer (3)

- Persistence (Stickiness)
  - "The Server" in OLG
  - How to handle information that must be kept across the multiple requests in a user's session.
- Session ID?
  - Cookie
  - IP Address
  - TCP Connection
- Pros & Cons ?

Server Load Balancer (4)

- SSL offloading (SSL/TLS termination)
  - Pros?
- Problems of Server Load Balancer
  - SPoF
  - Capacity Limit
  - Latency

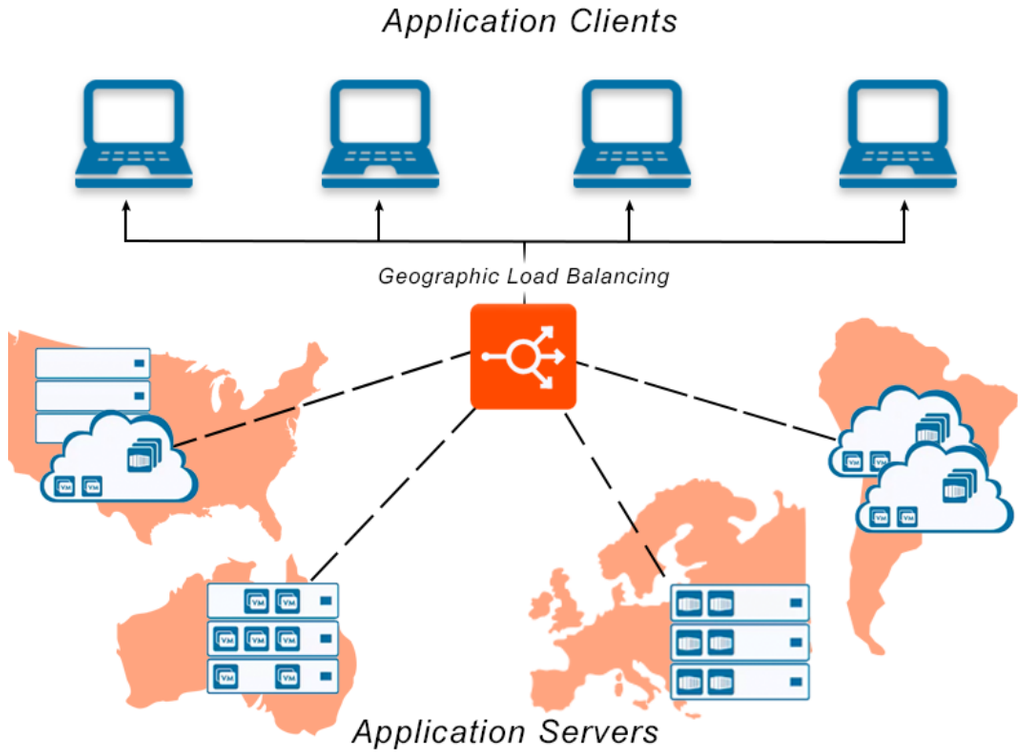


HW & SW of Server Load Balancer

- Nginx
  - Used in K8S
- PF in FreeBSD
- haproxy
- Envoy Proxy
- F5 BIG-IP
- A10
- on Cloud
  - AWS ELB (Elastic Load Balancer)
  - Google CLB (Cloud Load Balancer)

Global Server Load Balancer (GSLB)

- Globally balancing traffic to the nearest node.
- Pros
  - (Speed of light)
- Cons ?
- Technology
  - GeoDNS
    - resolve IP address based by the location of clients
  - Anycast
    - use BGP
    - Google DNS 8.8.8.8



Haproxy

- http://www.haproxy.org
- Reliable & High Performance TCP/HTTP Load Balancer
  - Layer 4 (TCP) and Layer 7 (HTTP) load balancing
  - SSL/TLS termination
  - Gzip compression
  - Health checking
  - HTTP/2

Haproxy - Installation

- In FreeBSD:
  - pkg install haproxy
  - You can also build it from ports
  - Config file: /usr/local/etc/haproxy.conf

Haproxy - Configuration

```
global
    daemon
    log 127.0.0.1 local0
    log 127.0.0.1 local1 notice
    maxconn 4096
    tune.ssl.default-dh-param 2048

defaults
    log               global
    retries           3
    maxconn           2000
    timeout connect   5s
    timeout client    50s
    timeout server    50s

listen stats
    bind 127.0.0.1:9090
    balance
    mode http
    stat enable
    stat auth admin:admin
```

Haproxy - Configuration

```
frontend www_csie_nctu
    bind 140.113.208.102:80
    mode http
    use_backend www_csie_nctu_server

frontend cscc_csie_nctu
    bind 140.113.208.103:80
    mode http
    use_backend www_csie_nctu_server

frontend game_server
    bind 140.113.208.104:9876
    mode tcp

backend www_csie_nctu_server
    balance roundrobin
    mode http
    http-request set-header X-forwarded-Port %[dst_port]
    http-request set-header X-forwarded-Proto https if { ssl_fc }
    server www1 192.168.99.1:80
    server www1 192.168.99.2:80
```

Haproxy - Configuration

```
backend cscc_csie_nctu_server
    balance roundrobin
    mode http
    option httpchk HEAD /health_check.php HTTP/1.1\r\nHost:\ cscc.cs.nctu.edu.tw
    option forwardfor
    http-request set-header X-forwarded-Port %[dst_port]
    http-request set-header X-forwarded-Proto https if { ssl_fc }
    server www1 192.168.99.101:80 check fall 3 rise 2
    server www1 192.168.99.102:80 check fall 3 rise 2
```

Haproxy Configuration

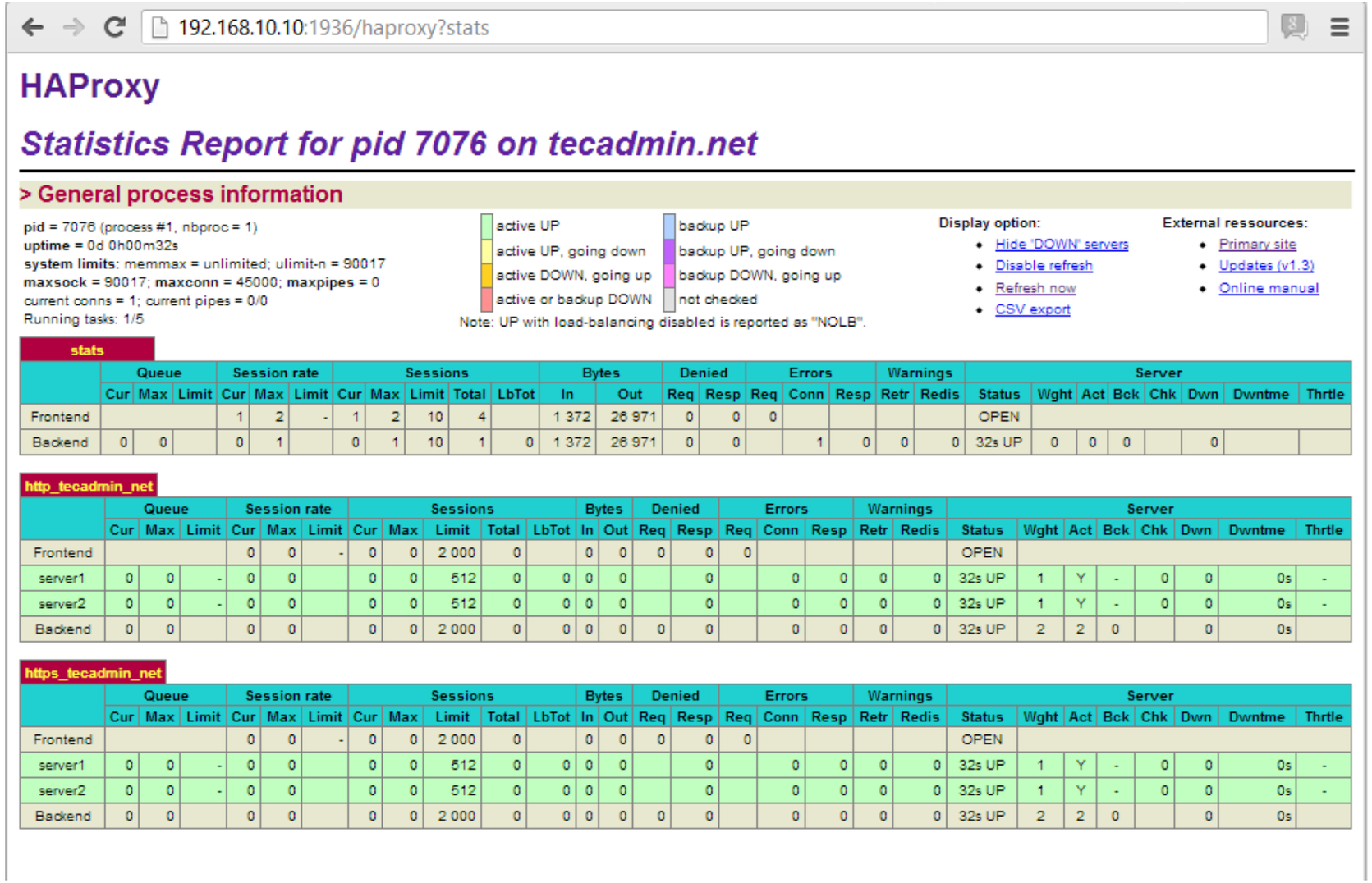
- global
  - log
  - chroot
  - uid / gid
  - pidfile

Haproxy Configuration

- defaults
  - log
  - option
  - retries
  - timeout

Haproxy Configuration

- listen
  - stats



Haproxy Configuration

- frontend
  - bind
  - mode
  - option
  - use_backend

Haproxy Configuration

- backend
  - balance
    - roundrobin, leastconn, hdr(param)
  - mode
  - http-request
  - server
    - check
    - fall
    - rise
    - inter
    - cookie

Haproxy - run

- /etc/rc.conf.local
  - haproxy_enable="YES"
- /usr/local/etc/rc.d/haproxy start

- Question: how to setup a backup node for haproxy?

Haproxy - Reference

http://cbonte.github.io/haproxy-dconv/2.1/configuration.html

Envoy

- https://www.envoyproxy.io
- Developed by Lyft (a ride-sharing company like Uber) and opensourced in 2017
  - Apache License 2.0
- Features
  - Dynamic APIs for configuration
  - Service Discovery
  - gRPC / MongoDB / HTTP support
- MicroService

Envoy - Installation

- Broken in FreeBSD now (require BoringSSL)
  - You can install it on Linux instead
- https://www.getenvoy.io
  - Debian: https://www.getenvoy.io/install/envoy/debian/
  - Ubuntu: https://www.getenvoy.io/install/envoy/ubuntu/
  - Centos: https://www.getenvoy.io/install/envoy/centos/

Envoy - Configuration

```
static_resources:
  listeners:
  - name: listener_0
    address:
      socket_address: { address: 127.0.0.1, port_value: 10000 }
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type":
type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManag
er
          stat_prefix: ingress_http
          codec_type: AUTO
          route_config:
            name: local_route
            virtual_hosts:
            - name: local_service
              domains: ["*"]
              routes:
              - match: { prefix: "/" }
                route: { cluster: some_service }
          http_filters:
          - name: envoy.filters.http.router
```

Examples — envoy 1.18.0-dev-fce386 documentation (envoyproxy.io)

Envoy - Configuration

```
clusters:
- name: some_service
  connect_timeout: 0.25s
  type: STATIC
  lb_policy: ROUND_ROBIN
  load_assignment:
    cluster_name: some_service
    endpoints:
    - lb_endpoints:
      - endpoint:
          address:
            socket_address:
              address: 127.0.0.1
              port_value: 1234
```

Examples — envoy 1.18.0-dev-fce386 documentation (envoyproxy.io)

Envoy - Configuration

- YAML file format
- Basic concept is same as haproxy
  - Listen (frontend) address
  - Backend addresses
  - Healthy Checks
    - https://www.envoyproxy.io/learn/health-check
  - Routes

Envoy - Run

- envoy -c config.yaml

Envoy - Reference

- https://www.envoyproxy.io/docs/envoy/latest/
- https://blog.getambassador.io/envoy-vs-nginx-vs-haproxy-why-the-open-source-ambassador-api-gateway-chose-envoy-23826aed79ef