Load Balancing and Scale-Out Applications

Ivan Pepelnjak (@ioshints, ip@ioshints.info)
NIL Data Communications

Scale Up or Scale Out?

Transactional databases

Web servers

The Problem: Broken TCP Stack

- IP addresses visible to the application layer
- Service = IP address + port# (in most cases)
- Multiple servers must be hidden behind a single address

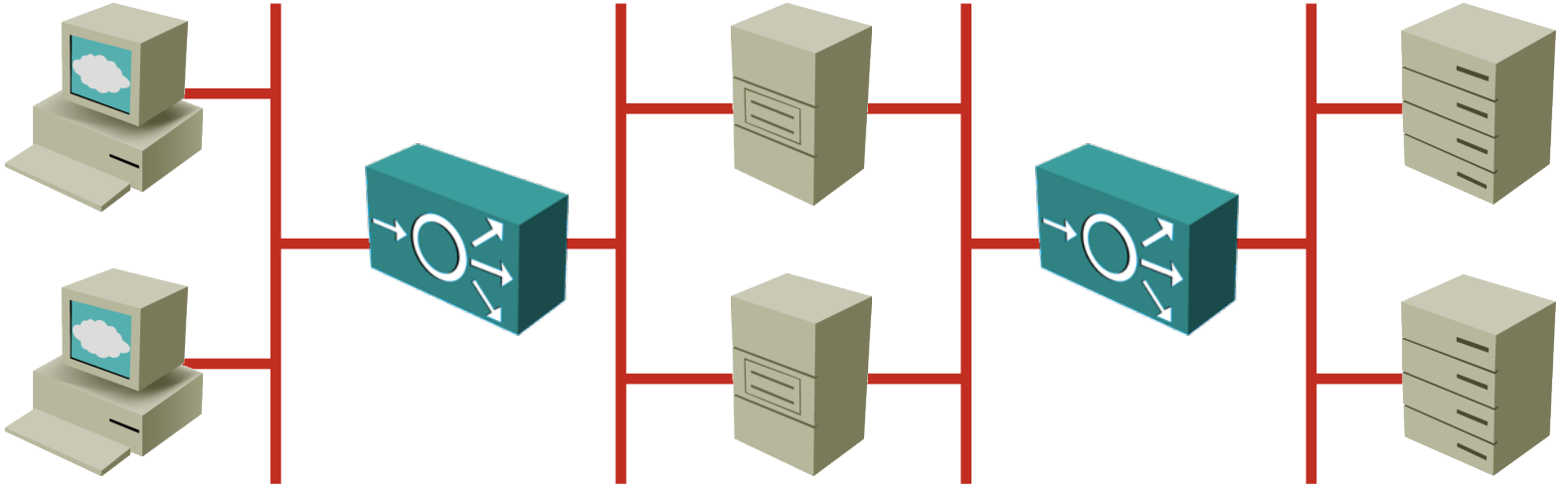
Solutions

- Load balancers
- Server clusters

Goals

- High availability
- Scalability

Load Balancing Solution Space

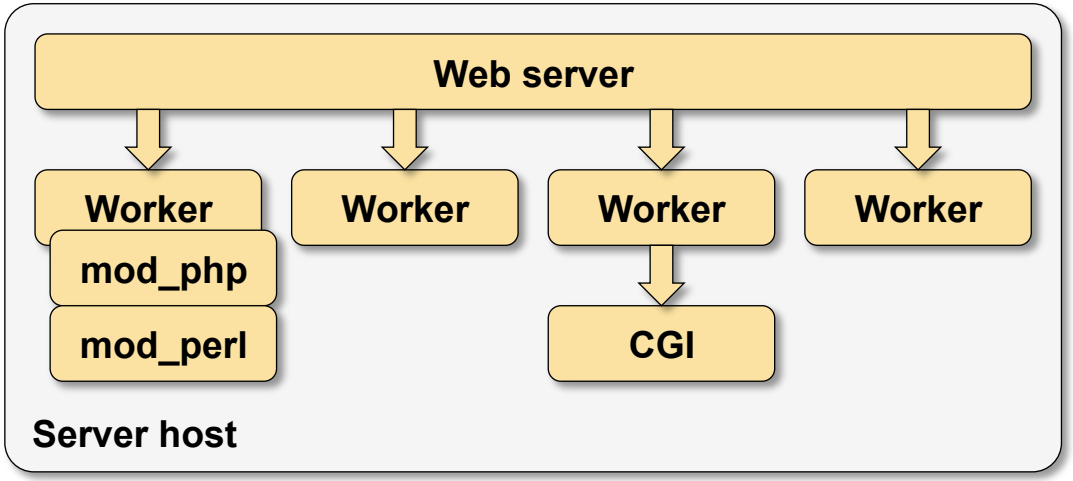


- Anycast — Unreliable
- DNS-based load balancing — Slow
- Dedicated load balancers — Expensive
- Server-based load balancing (server clusters) — Clumsy
- Application-based load balancing — Rare

Look Before You Jump

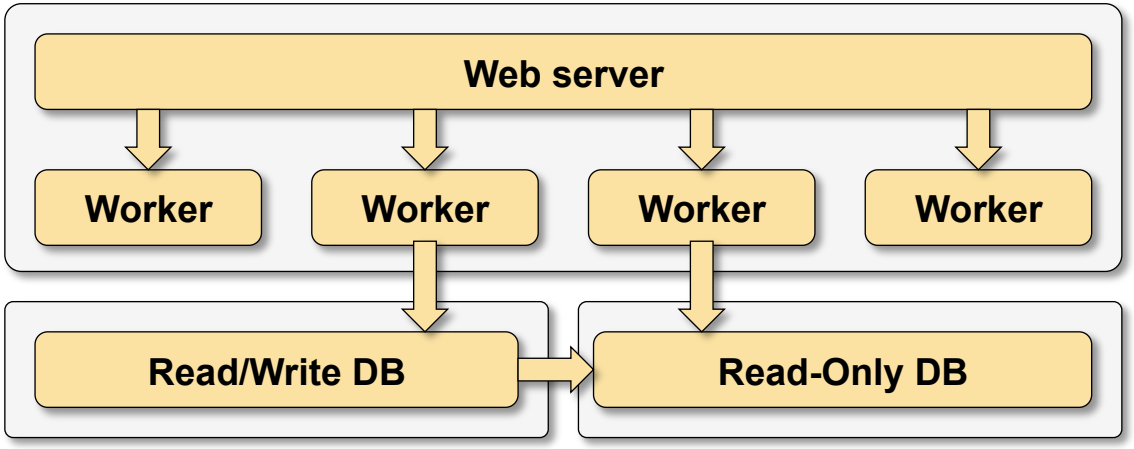
- Design application with scalability in mind
- Test a sample scale-out architecture (and failure handling)
- Deploy scale-out architecture when needed
- Investigate bottlenecks and fix application before deploying complex scale-out solutions

Web Servers: Worker Processes



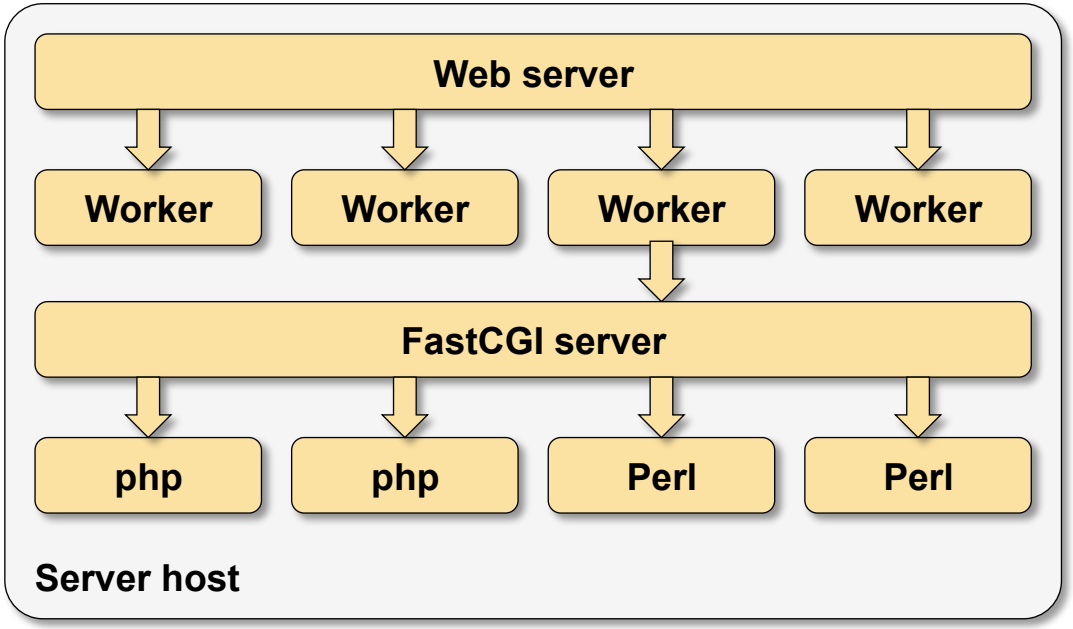
- HTTP requests served by worker processes (process fork)
- All worker processes are identical (and large)
- Scripts processed in worker processes or external programs (CGI)
- Client request blocks a worker process (or a thread)
- Persistent session occupies a worker process for a long time

Web Applications: Database Load Balancing



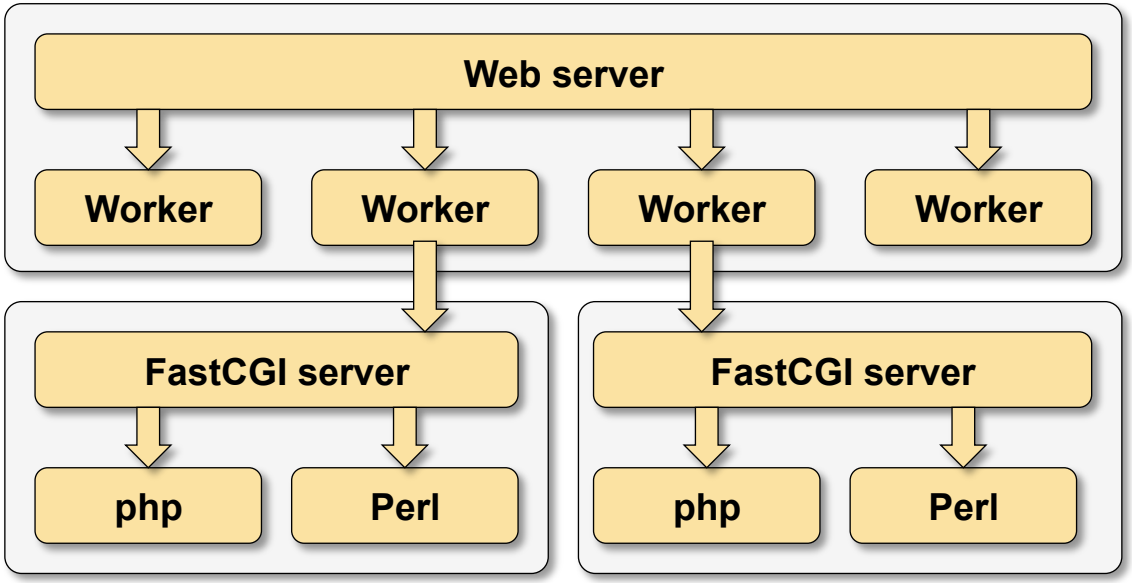
- Single R/W database replica and multiple R/O replicas
- Asynchronous replication (eventual consistency)
- Multiple database connections
- Most scripts access R/O replica(s)
- Solve per-user consistency issues with cookies

Web Servers: FastCGI



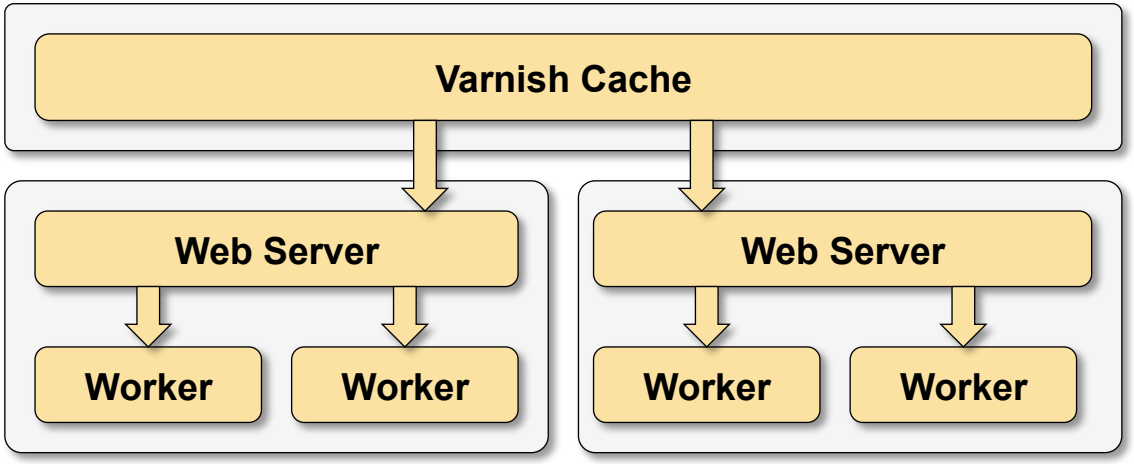
- Web server worker processes serve simple (static) requests
- Script processing offloaded to a different server
- Script output buffered in the worker process
- Client requests and persistent sessions no longer block script workers

FastCGI Offload and Load Balancing



- FastCGI works over TCP → you can separate web and app servers
- FastCGI server selection based on URL path → per-application servers
- FastCGI server selection bsaed on suffix → language-specific servers
- Multiple FastCGI servers (nginx, lighthttpd) → load balancing

Reverse Proxy and Load Balancing

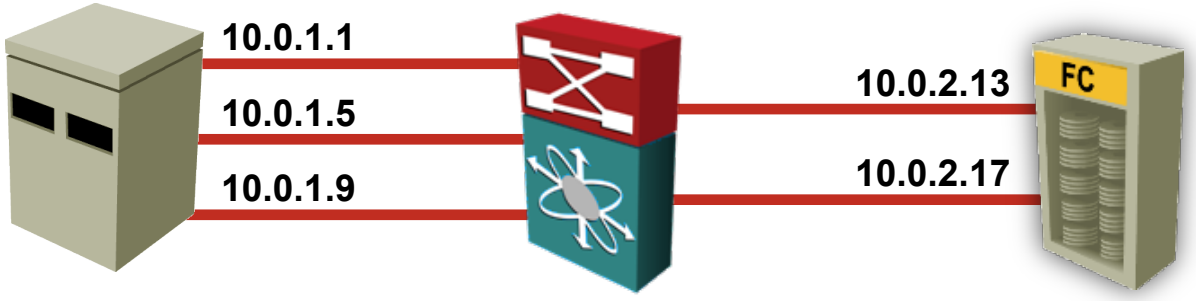


- Reverse proxy (front-end cache) can use multiple physical servers for a single HTTP hostname

Challenges:

- Load balancing mechanism
- Session persistence

Oracle with NFS

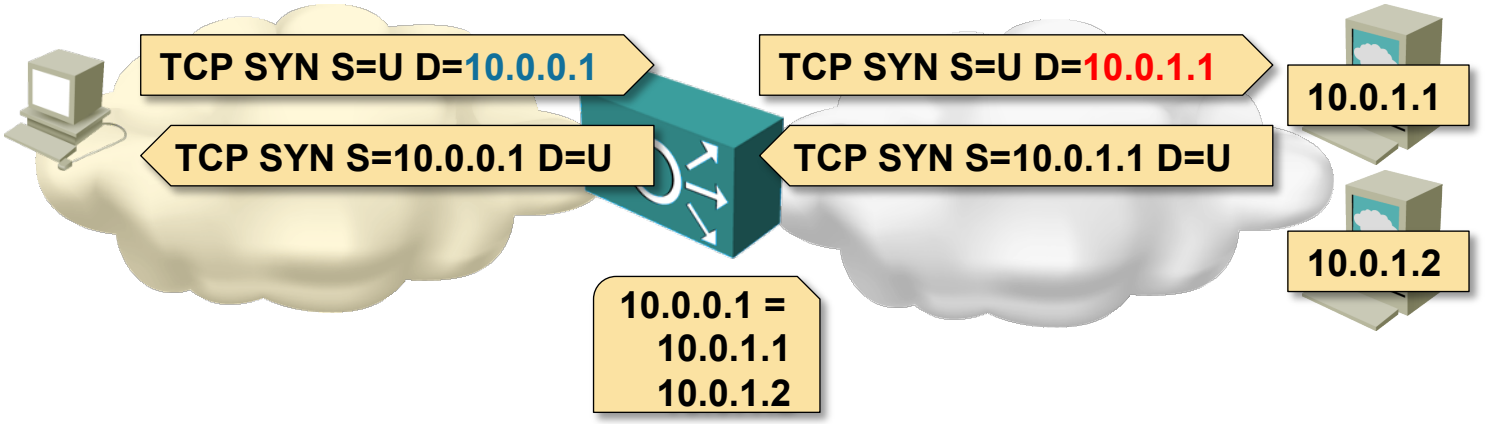


- NFS session is established from Oracle DB process
- Load balancing between multiple source and destination IP addresses

Other benefits

- Asynchronous I/O
- Reduced buffer copying overhead

Dedicated Load Balancers



Configuration

- Associate an *outside* IP address with a pool of *inside* IP addresses

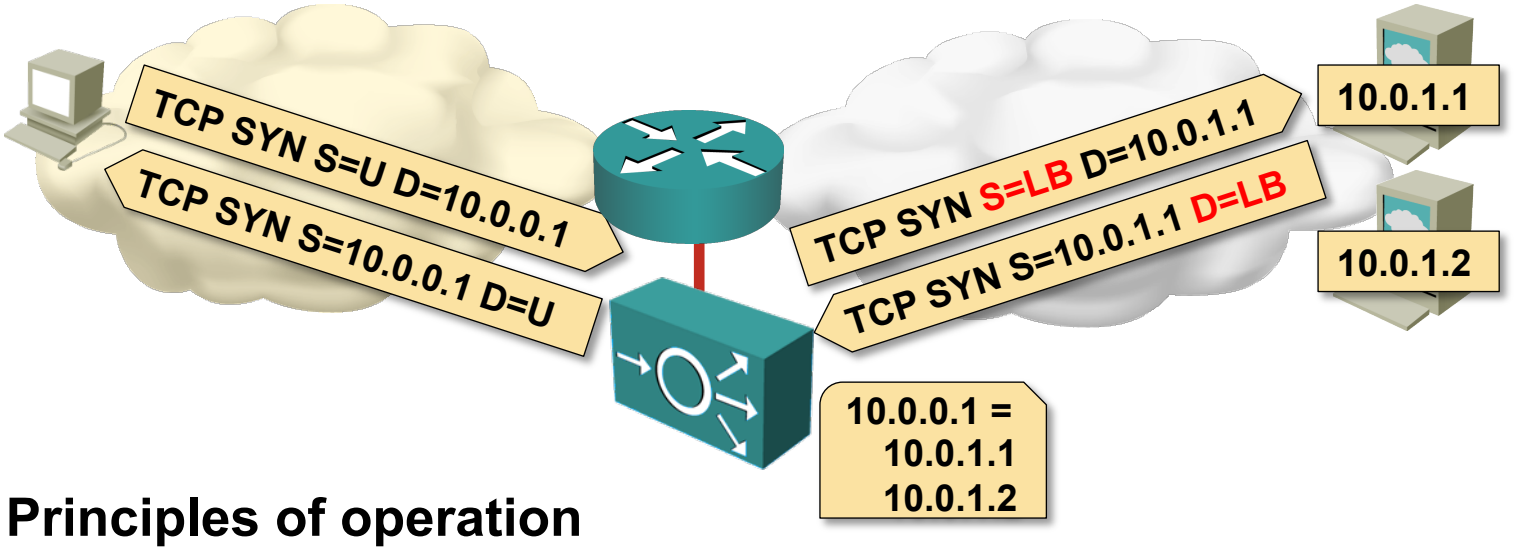
Control plane

- Monitor the health of *inside* servers
- Track the server load

Data plane

- Select the “best” *inside* server for a new session (incl. *stickiness*)
- Use NAT table to forward packets of existing sessions

Dedicated Load Balancers (One Arm Mode)



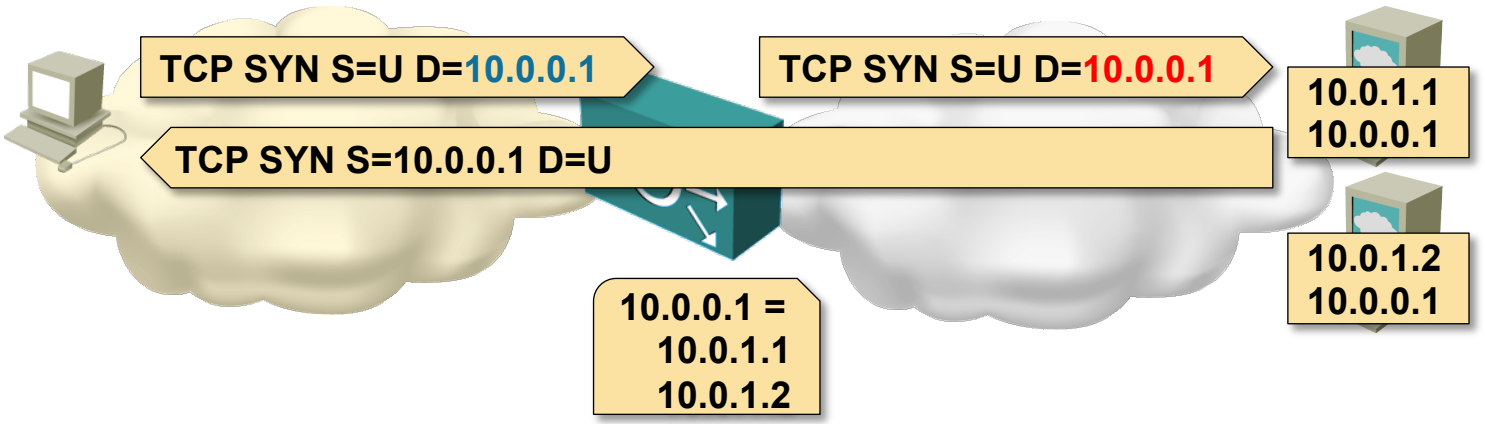
Principles of operation

- Both source and destination IP addresses are translated
- Client address is translated (NAT or PAT) into an address assigned to LB pool

Usage guidelines

- Use when the load balancer is not in the forwarding path
- Required for protocol translation (NAT64)
- Usually used with X-Forwarded-For HTTP header

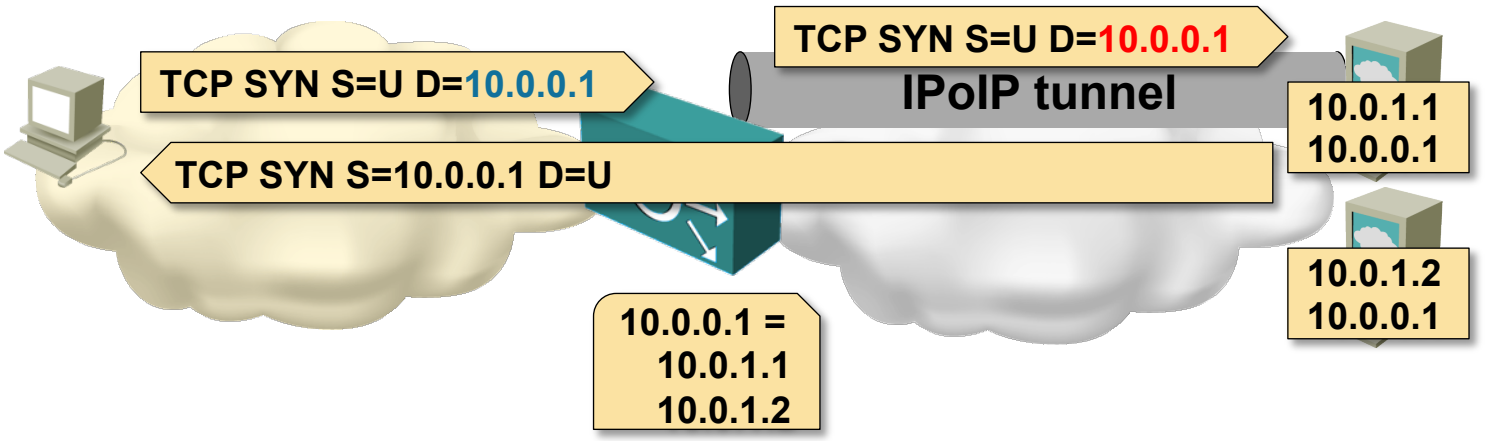
Direct Server Return



- Same IP address configured on all hosts (loopback interfaces)
- LAN IP address used for ARP (host MAC address resolution)
- Load balancer rewrites MAC header only
- Unmodified IP packet sent to selected server
- Server sends a reply packet directly to the client
- Requires L2 connectivity between load balancer and servers

Sample product: Linux Virtual Server (LVS)

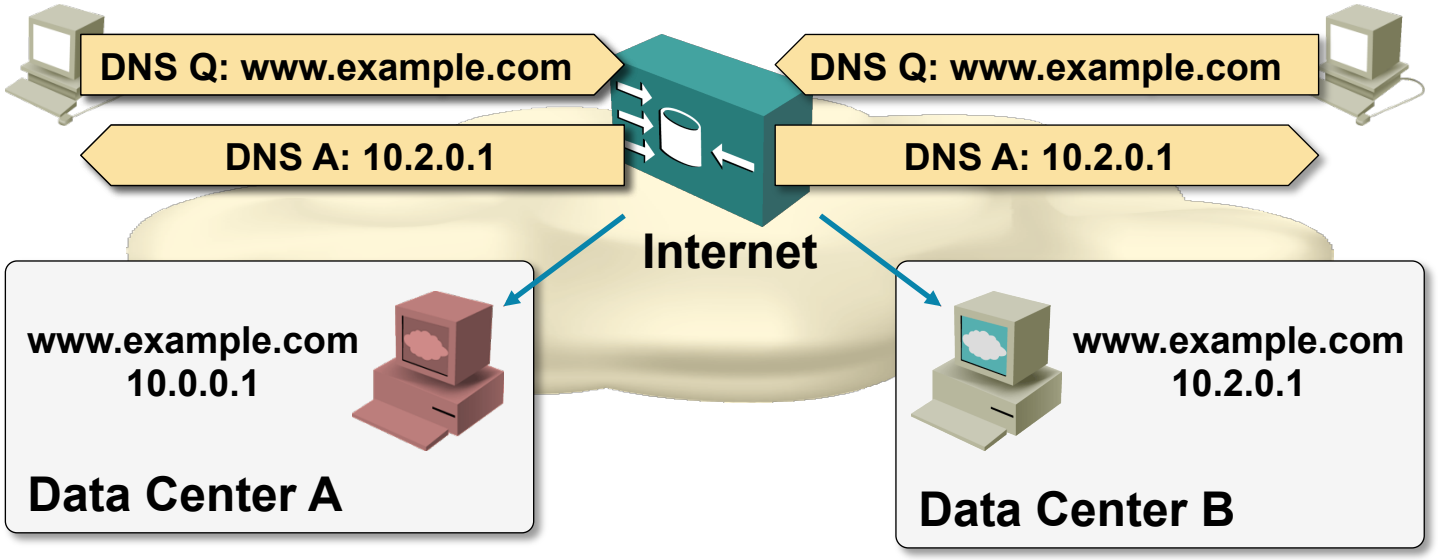
Direct Server Return with IP Tunnel



- Same IP address configured on all hosts (loopback interfaces)
- IP tunnels between load balancer and server(s)
- Load balancer encapsulates client IP packets
- Server sends a reply packet directly to the client
- Works with L3 connectivity between load balancer and servers

Sample product: Linux Virtual Server (LVS)

DNS-based Load Balancing



DNS responses vary based on user's location, server load and server availability

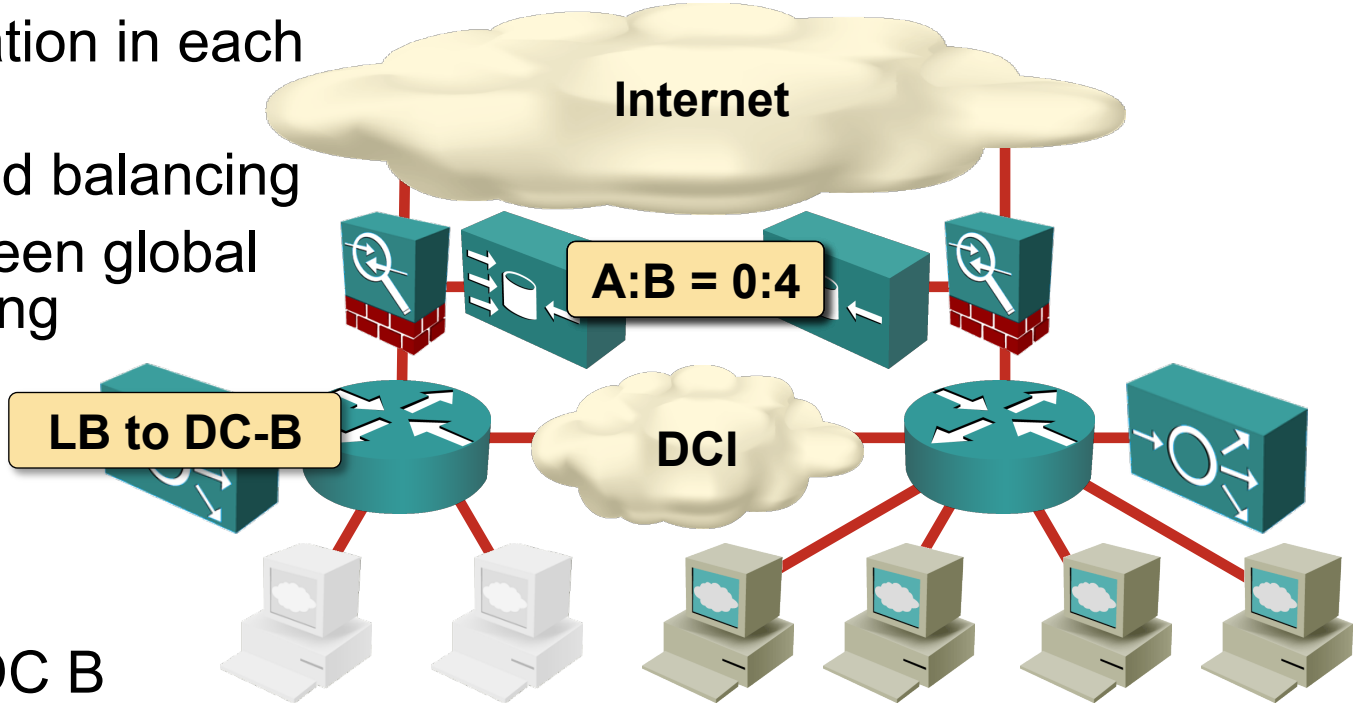
Caveats

- Geolocation based on recursive DNS server's location (not client's)
- Clients usually (but not always) pick the first IP address in the DNS response
- DNS pinning in browsers limits the usability of this solution

Disaster Avoidance With Load Balancing

Prerequisites

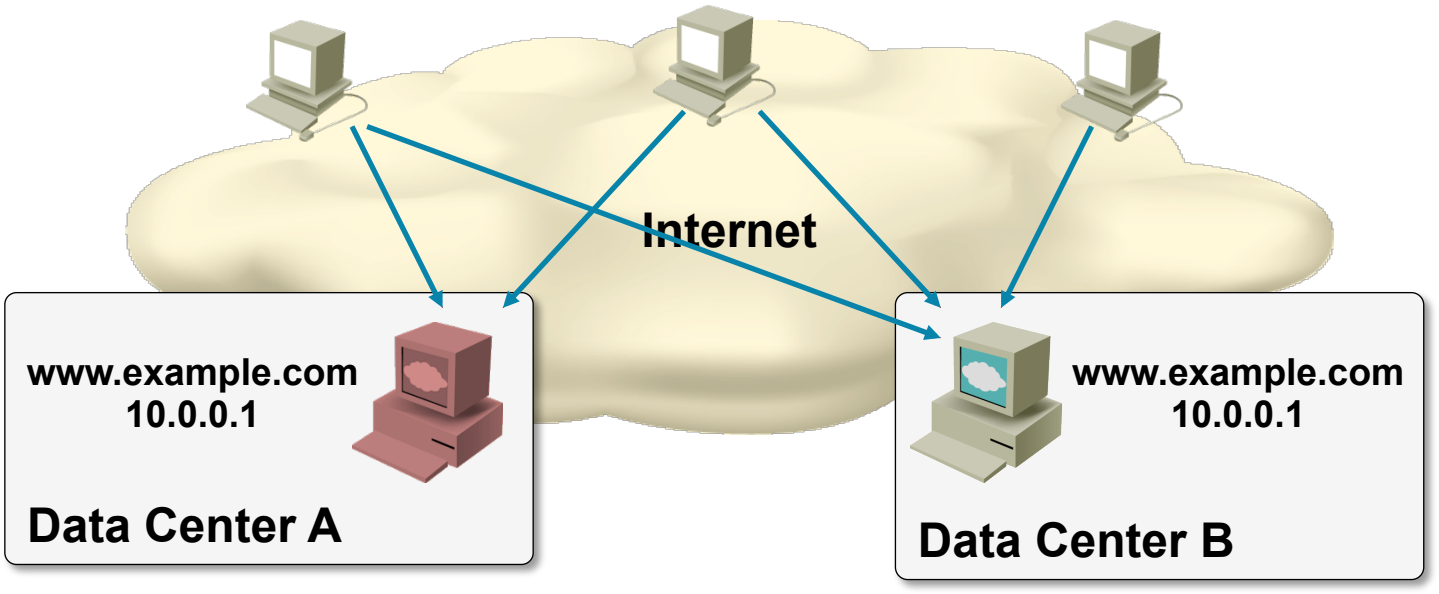
- Public VIP per application in each data center
- DNS-based global load balancing
- Synchronization between global and local load balancing



Process

- Graceful shutdown of servers in DC A
- Start new servers in DC B
- Load balancers shift load toward DC B
- No Layer-2 DCI or vMotion required

Anycast

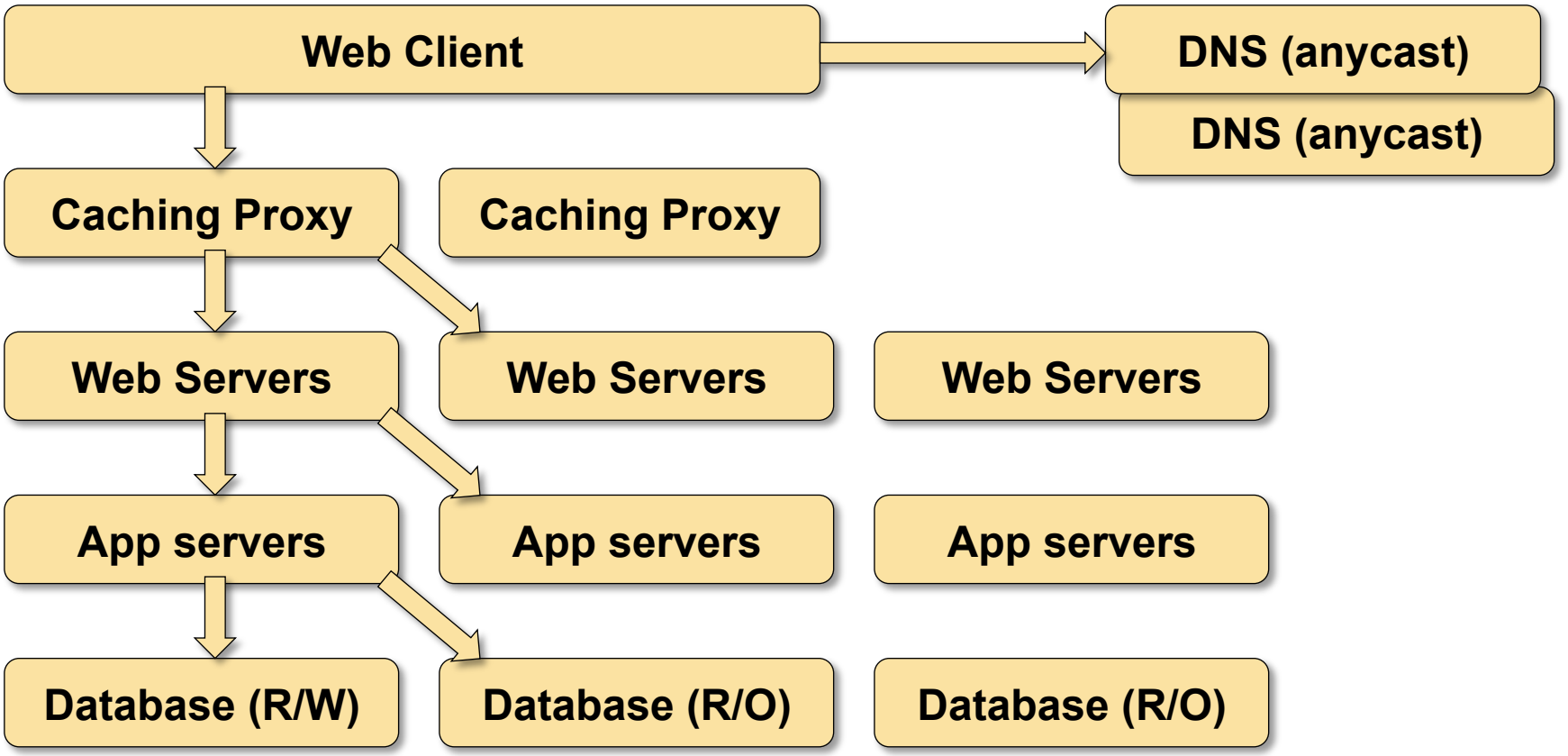


Same IP address is advertised from multiple data centers

Caveats

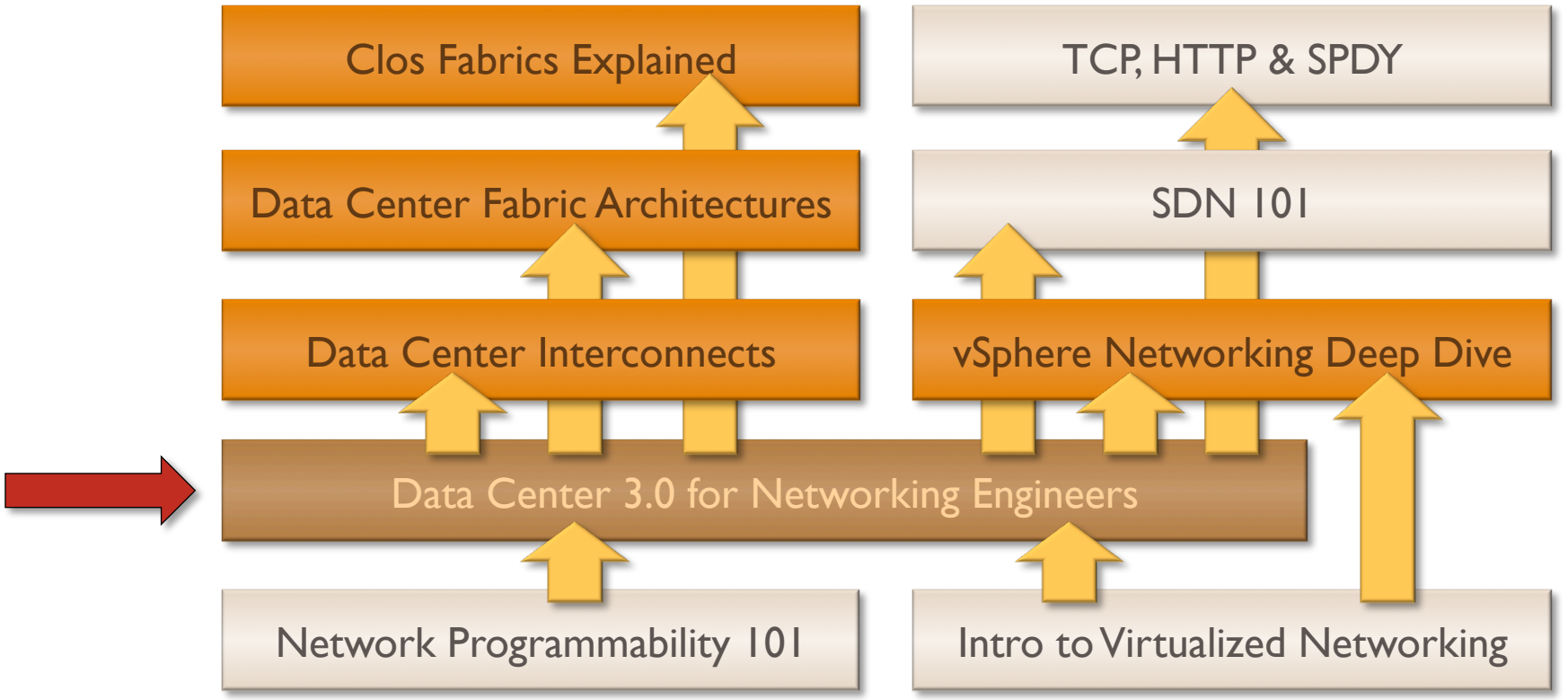
- Depends exclusively on Internet routing
- Perfect solution for UDP-based services (DNS)
- Quality of TCP-based services depends on network stability and routing distance between data centers

Scale-Out Application Architecture



- Still a few hotspots and single points of failure

More Information



Availability

- Live sessions
- Recordings of individual webinars
- Yearly subscription

Other options

- Customized webinars
- ExpertExpress
- On-site workshops

Questions?