

Infrastructure Technologies for Large-Scale Service-Oriented Systems

Kostas Magoutis

magoutis@csd.uoc.gr

<http://www.csd.uoc.gr/~magoutis>

Garage innovator

- Creates new Web applications that may rocket to popular success
 - Success typically comes in the form of “flash crowds”
- Requires load-balanced system to support growth
- Does not have access to large upfront investment

Contemporary utility computing

- Low overhead during lean times
- Highly scalable
- Quickly scalable

Storage delivery networks

- Amazon S3, Nirvanix platforms
- Similar to Content Delivery Networks (CDNs)
- Large clusters of tightly coupled machines
- Handle data replication, distributed consensus, load distribution behind a static-content interface

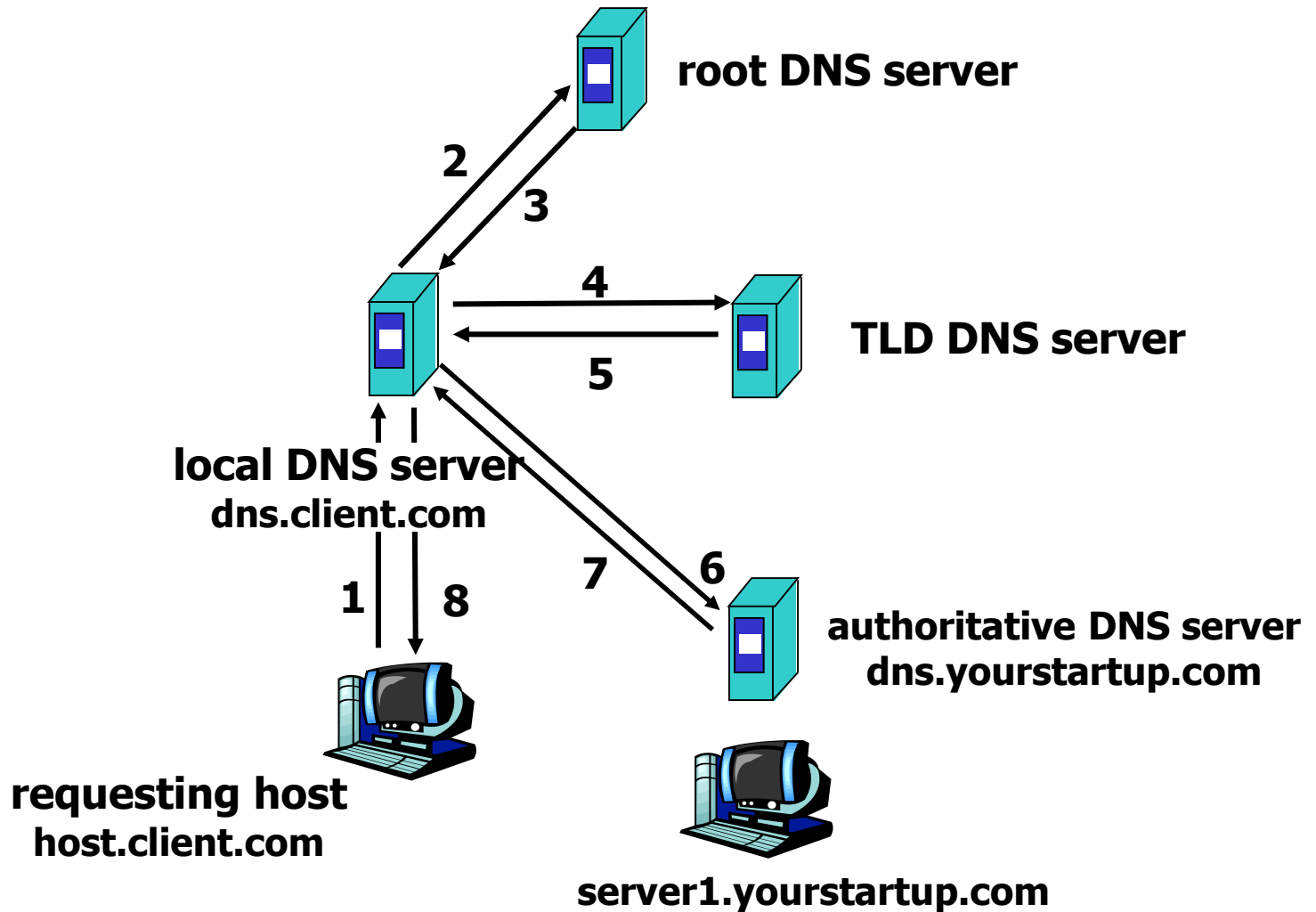
Compute Clouds

- Before Cloud computing (~2006):
 - Bandwidth to colocation facilities billed on per-use basis
 - Virtual private servers billed monthly
- Current utility computing providers offer VM instances billed per hour

Other building blocks

- Missing piece: relational databases
- DNS outsourcing
 - Avoids DNS becoming single point of failure

DNS example



DNS: caching and updating records

- Once any name server learns mapping, it caches it
 - Cache entries timeout after some time (TTL)
 - TLD servers cached in local name servers
 - Thus root name servers are not visited often
- update/notify mechanisms under design by IETF
 - RFC 2136
 - <http://www.ietf.org/html.charters/dnsind-charter.html>

DNS records

RR format: (name, value, type, TTL)

- Type=A
 - ❖ name is hostname
 - ❖ value is IP address
- Type=NS
 - **name** is domain (e.g. foo.com)
 - **value** is hostname of authoritative name server for this domain
- Type=CNAME
 - ❖ name is alias for some “canonical” (real) name
www.ibm.com is really
servereast.backup2.ibm.com
 - ❖ value is canonical name
- Type=MX
 - ❖ value is name of mail server associated with name

Inserting records into DNS

- Example: just created startup “Network Utopia”
- Register name networkutopia.com at a registrar (e.g., Network Solutions)
 - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
 - Registrar inserts two RRs into the com TLD server:
 - (networkutopia.com, dns1.networkutopia.com, NS)
 - (dns1.networkutopia.com, 212.212.212.1, A)

Inserting records into DNS (2)

- Put in authoritative server Type A record for `www.networkuptopia.com`
- Put Type MX record for `networkutopia.com`

Scaling architectures

- Using the bare SDN
- DNS load-balanced cluster
- HTTP redirection
- L4 or L7 load balancing
- Hybrid approaches

Analysis of the design space

- Application scope
- Scale limitations
- Client affinity
- Scale up/down time
- Response to failures

Application scope

- Bare SDN suitable for static content only
- HTTP redirector works with HTTP
- L7 load balancers constrained by application protocol
- DNS and L4 load balancers work across applications

Scale limitation

- SDNs are designed to be scalable
- HTTP redirection involved only in session setup
- L4/L7 load balancer limited by forwarder's ability to handle entire traffic
- DNS load balancing has virtually no scalability limit

Client affinity

- SDN fulfills client request regardless of where it arrives
- HTTP redirection provides strong client affinity
 - Use client session identifier
- L4 balancers cannot provide affinity
- L7 balancers can provide affinity
- DNS clients cannot be relied upon to provide affinity

Scale up and down time

- Bare SDN designed for instantaneous scale up/down
- HTTP redirectors and L4/L7 balancers have identical behavior
 - Scale down time is trickier, need to consider worst-case session length
- DNS is most problematic

Effects of front-end failure

- SDN has multiple redundant hot-spare load balancers
- L4 and L7 balancers are highly susceptible
 - A solution is to split traffic across m balancers, use redundant hot spares (DNS load-balanced)
- HTTP redirectors same as above, except that there is no impact on existing sessions
- DNS load balancing affected by failure when
 - Using single DNS server (no replication)
 - Short TTLs so as to handle scale-up/down and backend node failure

Effects of back-end failure

- “Back-end” are servers that are running service code
- SDN managed by service provider ($\sim 1\%$ writes fail)
- HTTP redirector and L4/L7 balancer
 - Newly arriving sessions see no degradation at all
 - Existing sessions see only transient failures
- DNS load balancing suffers worst performance

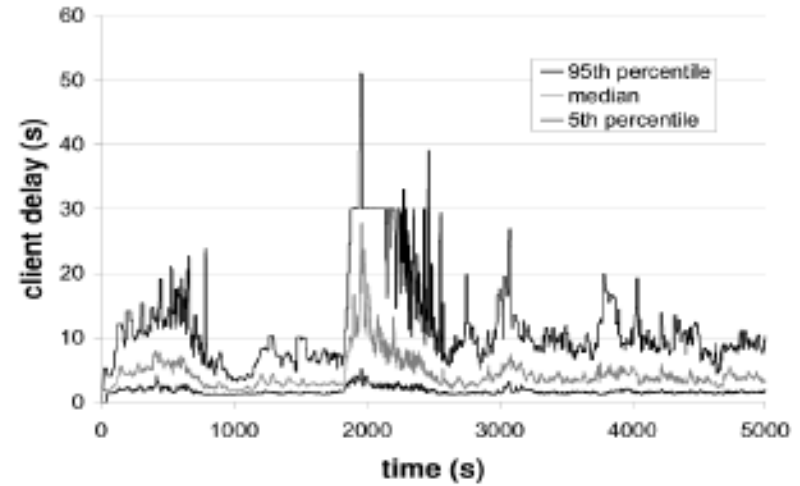
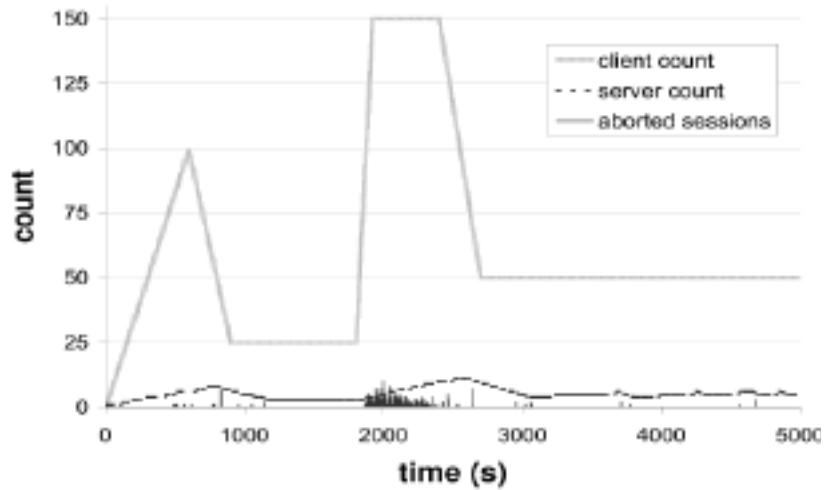
Summary

Criterion	Design			
	Bare SDN	HTTP Redir.	L4/L7 Load Bal.	DNS Load Bal.
§3.1: Application Scope	Static HTTP	HTTP	All	All
§3.2: Scale Limitation	Very large	Client arrival rate	Total traffic rate	Unlimited
§3.3: Client affinity	N/A	Consistent	Consistent	Inconsistent
§3.4: Scale-Up Time	Immediate	VM Startup Time (about a minute)	VM Startup Time (about a minute)	VM Startup + DNS TTL (5-10 minutes)
§3.4: Scale-Down Time	Immediate	Session Length	Session Length	Days
§3.5: Front-End Node Failure: Effect on New Sessions	N/A	Total Failure	Total Failure	Major Failure
§3.5: Front-End Node Failure: Effect on Estab. Sessions	N/A	No effect	Total Failure	Rare effect
§3.5: Front-End Node Failure: Effect on New Sessions (<i>m</i> redundant front-ends)	Unlikely	long delay for $1/m$ th sessions?	long delay for $1/m$ th sessions?	Short delay (§4.2)
§3.5: Front-End Node Failure: Effect on Estab. Sessions (<i>m</i> redundant front-ends)	Unlikely	No effect	$1/m$ th sessions fail	A few sessions see short delay
§3.6: Back-End Node Failure: Effect on New Sessions	Unlikely	No effect	No effect	long delay for $1/n$ th of sessions
§3.6: Back-End Node Failure: Effect on Estab. Sessions	Unlikely	User-recoverable failure	Transient failure	long delay for $1/n$ th of sessions

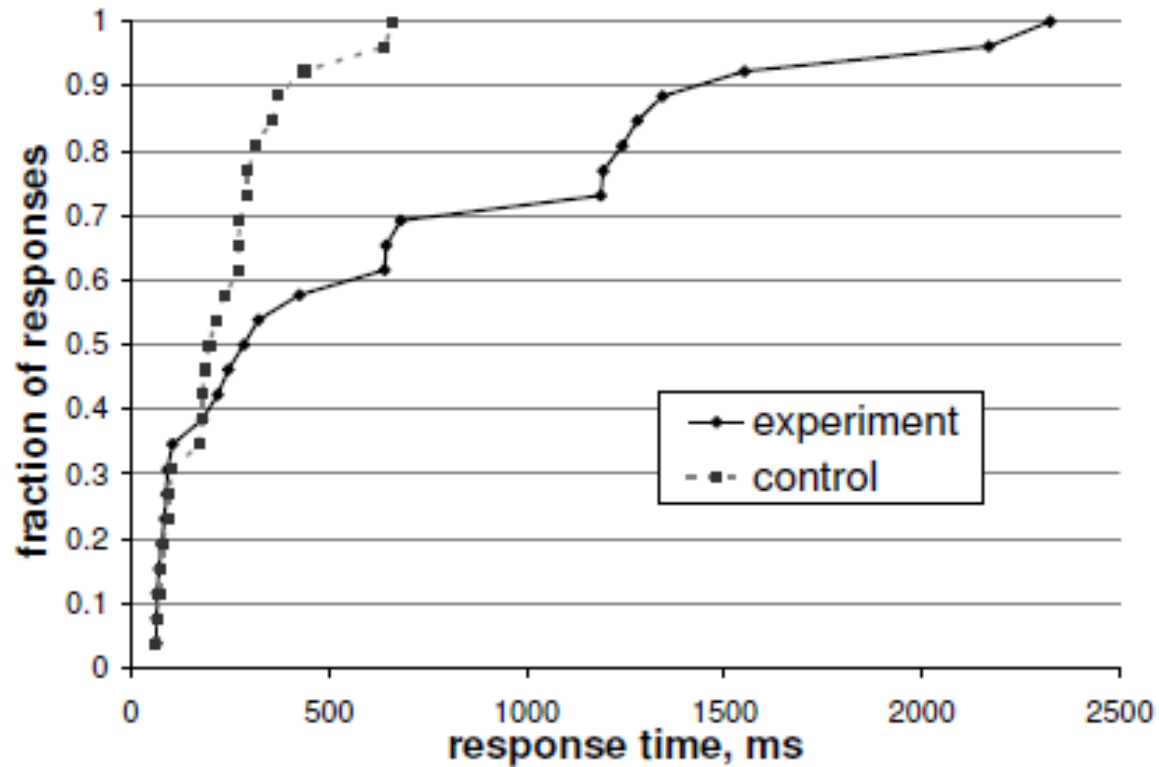
EC2-integrated HTTP redirector

- Monitors load on each running service instance
 - Servers send periodic heartbeats with load statistics
 - Redirector uses heartbeats to evaluate server liveness
- Resizes server farm in response to client load
 - When total free CPU capacity on servers with short run queues are less than 50%, start new server
 - When more than 150%, terminate server with stale sessions
- Routes new sessions probabilistically to lightly loaded servers

HTTP redirect experiment



DNS server failover behavior



Other microbenchmarks

- Web client DNS failover behavior
 - Clients experience delays from 3 to 190 seconds
- Badly-behaved resolvers
- Maximum size of DNS replies
- Client affinity observations

MapCruncher

- Interactive map generated by client (AJAX) code
- Service instance responds to HTTP GET bringing an image off of stable storage
- Initially used 25GB of images on a single server's disk
- Flash crowd service peaked at 100 files / sec
- Moving to Amazon S3 solved I/O bottleneck

Asirra

- CAPTCHA Web service
- Asirra session consists of
 - Client retrieves challenge
 - Submits user response for scoring
 - Produce service ticket to present to webmaster
 - Webmaster independently verifies service ticket
- Deployed in EC2
 - 100GB of images (S3)
 - Metadata (MySQL) reduced into simple database loaded on each server's local disk

Asirra (2)

- Session state kept locally within each server
 - S3 option considered inadequate (write performance)
- Client affinity becomes important
 - DNS load balancing does not guarantee affinity
- Servers forward session to its home
 - Rate of affinity failures about 10%
- Flash crowd
 - 75,000 challenges plus 30,000 DoS requests over 24 hours

Asirra lessons learned

- Poor client-to-server affinity due to DNS load balancing was not a big problem
- EC2 lost IP reservation after failure (fixed)
- Denial of service attack easily dealt with with Cloud resources
 - Further lesson: No need to optimize code before on-going popularity materializes

Inkblot

- Website to generate images as password reminders
 - Must store dynamically created information (images) durably
- Coded simply but inefficiently in Python
- Store both persistent and ephemeral state in S3
- Initial cluster consistent of two servers, load balanced through DNS
 - Updating DNS required interacting with human operator

Inkblot (2)

- Flash crowd resulted into run-queue length of 137
 - Should be below 1
- Added 12 more servers, DNS update, within half hour
- New server saw load immediately, original servers recovered in about 20 minutes
- 14 servers averaged run queue lengths b/w 0.5-0.9
- After peak, removed 10 servers from DNS, waited an extra day for rogue DNS caches to empty