Scaling Web Systems

Dad, what are clouds made of?

Linux servers, mostly.
Review: Systems, Threads, Sockets

- Operating systems use **process** structures to manage program execution
  - Processes have isolated address space
- Processes can create and manage **threads** to execute multiple things in parallel
  - UI thread, networking thread, heartbeat thread, audio thread, etc.
  - Threads share the same address space as the host program
- **Threads** are subject to **race conditions** that can be managed with **synchronization primitives** like **locks**
- Networked applications use **sockets** to manage communication with one another
  - The OS manages sockets across many applications
One Slide Summary: Scaling Web Systems

- Google-scale systems must handle millions of queries per day
  - We also want to survive distributed denial of service (DDoS) attacks
- We can use DNS to help scale access to servers
  - One hostname can resolve to multiple IP addresses to balance load
- Load balancing is the act of distributing requests across servers
- By replicating servers, no single device needs to handle all requests
- We use sharding to distribute storage across multiple servers
  - A database may consist of multiple shards to help distribute access
- Replicas and shards must communicate to remain consistent
(Total) Recall: Server-side Dynamic Content

- **Client** specifies a URL
  - **Server** runs a *function*, serves returned *output*

- How does function generate content?
  - Maybe it checks some *session* information
  - Probably a few *DB* queries in there
  - **Renders** a template down to HTML/CSS
  - **Serves back** a big string

- Generation of content specific to each request
  - REST API is server-side dynamic content, too (generating new JSON...)

GET READY FOR A SURPRISE
Tiers of Web Services

• So far: one web server (e.g., Flask), and one database server (SQLite)
• If we want scale, we can think about managing replicas for each layer of the web service
  • Web servers
  • API servers
  • Database servers
  • Asset servers...
Replication

• Scaled implementation: replicate web servers and database servers
Replication

- Client selects among web servers
- Web server selects among database servers
Web server replication

• For one request, how do we select one web server?
Web server replication
Connect client to one web server

• How does a client select one web server?
  • Round robin DNS
  • Load balancer (proxy)
  • Both
Round robin DNS

• Multiple IP address for one domain name
• DNS server responds to a DNS request with a *list of IP addresses*

$ host google.com

google.com has address 192.122.185.23
google.com has address 192.122.185.34
google.com has address 192.122.185.53
google.com has address 192.122.185.59
...

Round robin DNS

• Permute the list with each response
• No set method for client to select address
  • Usually just the first address
• Does not consider
  • Transaction time
  • Server load
  • Network congestion
  • Etc.
• No protection from server going down
More Generally: Planned DNS

• The DNS server produces **IP addresses** in response to **hostnames**
  • We can exploit this for distributing load!

• Recall: IP addresses can convey geographic information

• If your DNS server receives a request to resolve “google.com”:
  • If the requester’s IP is near Chicago, return the closest replica!
  • (google kind of does this already – remind Kevin to demo)
Load balancing

• Alternative to round robin DNS: load balancing
• Load balancer, AKA proxy
• "Middleman" forwards requests to backend servers
• End user doesn't need to know about them
  • Load balancing is transparent to the end user
    • They don’t care about specific machine they’re connecting to, they just want the service to work

• TL;DR a load balancer is a network appliance that forwards traffic to multiple machines
Load balancing
Load balancing
Load balancing
Round Robin DNS and Load Balancing

• Large websites use both

• Round Robin DNS to multiple load balancers for redundancy

• Load balancers to backend web servers for scaling

• Web servers “should” be stateless
  • That's what the database is for!
  • Alternative: load balancer can decide based on the session id
Database replication

Eventually you’ll need more database servers
Database replication

• For one request, how do we select one database server?
Database replication

**master (AKA primary)**

```
1 INSERT INTO my_table VALUES (...)
2 DELETE FROM my_table WHERE ...
  ...
```

**replica**

Replication
Read example

master (AKA primary)

replicas

GET

/search?
find_desc=Pizza
&find_loc=Ann+Arbor%2C+MI
&ns=1
Write example

replicas

master (AKA primary)

replicas

POST

-- INSERT INTO reviews VALUES (...) --

Your Next Review Awaits

Monash's Seafood Market

Great Salmon teriyaki!

Share your review on: Facebook, Twitter

Save Now

Post Review

MySQL

MySQL

MySQL

MySQL
Database replication

• Replicate the data so any DB can handle any read query
• Good for read query workload

• Bad for updates/insertions
  • Remember consistency and definition?

Me: *clears off table after dinner*

Database team:
Partitioning Data into Shards

• A **shard** is a server that contains a **partition** or **fraction** of data

• When a request comes in, it must be routed to the appropriate shard

• Very well-suited for cleanly-partitioned data
  • e.g., break down student database by starting letter of uniqnames

• Database requiring **merging** of shards is a nightmare
Thought Questions

• What kind of websites might need:
  • more front-end servers than database servers?
  • more database servers than front-end servers?

• Which of sharding or replication would be better for scaling databases that store the following data?
  • Edits to Wikipedia articles
  • Tweets on Twitter made by different users
  • Airline flight reservation system
Wide area distribution

• Orthogonal to replication/partitioning
• Replicas called "mirrors"
• Good for disaster recovery
• Mirroring can even be used for web server
  • Exploit network proximity
• Partitioning can often be aligned geographically
  • E.g. account based on user geo-location
  • And not too much shared info across accounts
Wide area distribution
Caching

• Keeping a local copy of a resource
  • DNS Resource Records
    • IP addresses
  • Static page content
    • HTML, images, JS source code
  • Dynamic page content
    • If it doesn’t change very often

• Why? Fewer network requests, lower latency, fewer hops, higher bandwidth
Multiple points of caching

- Multiple places we cache in the web
  - Browsers
  - Proxies
  - Edge (Akamai)
  - Servers ("front-side caching")
- Each independently decides what to keep in cache
- Cache space is limited
  - Choice is an optimization problem
  - Policy solved independently by every application
Summary So Far

• DNS Planning, Replication, Load Balancing, Sharding, and Caching all help achieve **scale** in web services
  • DNS: One host, multiple IP addresses
  • Replication: Copy assets to multiple places
  • Load balancing: distribute network requests among replicas
  • Sharding: divide database among multiple servers
  • Caching: keep fast copies of data nearby

• But how do we get a server? Where are they?
Web physical infrastructure

• We've talked about scaling dynamic content so far
• Replicate web servers
• Replicate database servers

• Where is all this physical infrastructure?
• Where is the web?
The data center

• The internet connection physical infrastructure on the last few slides connect physical machines
• Those machines have to go someplace!
### Data center functions

<table>
<thead>
<tr>
<th>Security</th>
<th>Biometric scan, locked doors</th>
<th>Guard, cards, faraday cages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>A/C!</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>Multiple Physical Taps</td>
<td>Multiple Networks</td>
</tr>
<tr>
<td>Power</td>
<td>Transformers</td>
<td>Diesel, batteries, flywheels</td>
</tr>
<tr>
<td>Weather hardening</td>
<td>Roof!</td>
<td>Earthquake resistance</td>
</tr>
</tbody>
</table>
Data center economics

• Some data centers run by a single entity
  • Google, Microsoft, University of Michigan

• Others rent space to smaller customers

• Cost of running data center combination of:
  • Amortized costs of the building (15 yr)
  • Amortized costs of electrical equipment
  • Amortized costs of servers (3 yr)
  • Power, Bandwidth, Labor

• When you rent space, you pay for:
  • Space, power, bandwidth, maybe labor
Data center energy growth

• Very old data, but... installed base grows 11%/year
• In 2012, 2% of all US energy use
Heat

• Servers account for barely half of power
  • 1W of cooling per 1.5W of IT load
• Managing energy consumption means, to a large extent, managing heat
Cooling the data center
Cooling the data center
Cooling the data center
Cooling the data center
Evaporative cooling
Other sources of inefficiency

• Energy wasted by idle systems
  • Up to 75% idleness; but idle power ~60% of peak

• Inefficient cool air distribution
  • Hot exhaust recirculation halves cooling efficiency

• Power conversion losses
  • 4+ conversions; some under 70% efficiency

• Poor server performance per watt
  • Embedded sys. up to 5x better on Web 2.0 apps
Data center efficiency

• Power Usage Effectiveness (PUE)
  • Total Facility Power / IT Equipment Power
  • For each watt, how much to computing?
  • 1.0 means no extra cost at all
  • Facebook reports 1.07 for new Fort Worth data center
  • Old Michigan data center is ~2.0, but others on campus are 1.1
Current challenges

• Whole data center must be optimized, not just filled with more efficient computers
  • CPUs only account for 12% of energy!
• Challenges:
  • Power infrastructure very inefficient
  • Utilization & poor energy-proportionality
  • Cooling efficiency
• Green energy (e.g., wind farms)
Utilization

• Machines usually very poorly utilized
Utilization

• Why so poorly utilized?
  • Work spread over many machines for robustness, data safety
  • Natural variance in load means most times will not be peak
    • Is it ever possible to do more work during off-peak times to reduce work during peak times?
      • Often, no
  • Server-class machines often mismatched to Web workloads
  • Many background tasks mean machines never completely idle
Cloud Computing: Let’s Share Computers

- The **cloud** refers to a collection of practices used to increase utilization and decrease costs by offering computing services **on-demand**

**Idea:** Cloud Provider has powerful rack servers and infrastructure
- Users (companies) “rent” CPU time, or RAM, or storage, from Cloud Provider
- Users pay for what they need, no more, no less
- Need a webserver? Buy a chunk of the Cloud Provider’s servers

**Benefits:** Each “tenant” may have low utilization, but with multiple “tenants”, Cloud Provider has high utilization overall
Cloud services have taken over

• Most web services aren’t run on dedicated hardware anymore

Netflix finishes its massive migration to the Amazon cloud

After move to Amazon, only the DVD business still uses traditional data center.

When Amazon’s cloud storage fails, lots of people get wet

By MAE ANDERSON, AP TECHNOLOGY REPORTER
NEW YORK — Feb 28, 2017, 7:50 PM ET
Cloud services have taken over

• They’re run on Amazon Web Services, Google Cloud, or Microsoft Azure
What IS the cloud?

• Unspoken: we don’t really agree on a definition
• But increasingly, it refers to compute services offered on hosted hardware and software
• Amazon, Microsoft offer a slew of services *for rent*
  • On-demand virtual machines
  • On-demand storage
  • On-demand GPUs
What IS the cloud?

• Unspoken: we don’t really agree on a definition
• But increasingly, it refers to compute services offered on hosted hardware
• Amazon offers incredible array of services for rent

| Amazon Athena | Amazon EC2 Container Service (ECS) |
| Amazon API Gateway | Amazon EC2 Systems Manager |
| Amazon AppStream | Amazon ElastiCache |
| Amazon AppStream 2.0 | Amazon Elastic Block Store (EBS) |
| Amazon Chime | Amazon Elastic Compute Cloud (EC2) |
| Amazon Cloud Directory | Amazon Elastic File System (EFS) |
| Amazon CloudSearch | Amazon Elastic MapReduce |
| Amazon CloudWatch | Amazon Elasticsearch Service |
| Amazon CloudWatch Events | Amazon Elastic Transcoder |
| Amazon CloudWatch Logs | Amazon GameLift |
| Amazon Cognito | Amazon Glacier |
| Amazon DynamoDB | Amazon Inspector |
| Amazon EC2 Container Registry | Amazon Kinesis Analytics |
| Amazon Kinesis Firehose | Amazon Kinesis Streams |
| | Amazon Lightsail |
| | Amazon Machine Learning |
| | Amazon Mobile Analytics |
| | Amazon Pinpoint |
| | Amazon Polly |
| | Amazon QuickSight |
| | Amazon Redshift |
| | Amazon Rekognition |
| | Amazon Relational Database Service (RDS) |
| | Amazon SimpleDB |
| | Amazon Simple Email Service (SES) |
| | Amazon Simple Notification Service (SNS) |
| | Amazon Simple Queue Service (SQS) |
Core Amazon Web Services (AWS)

• Two services are the most important for us:
  • *Elastic Compute Cloud* ("EC2") offers machines for rent
  • *Simple Storage Service* ("S3") offers very basic Web storage

• An EC2 “instance” with 16 CPUs, 64GB costs $0.80 an hour.
  • That’s $7,008 a year!
  • Dell will sell me something similar for ~$4000.
  • Why would I ever use AWS?
Advantages of Infrastructure-as-a-Service

• Pay only for what you consume
  • LOTS of different sized machines
  • You don’t have to overcommit for resources
  • Term-limited: If your services needs to stop, you just stop paying

• Economies of scale: Cloud Provider manages racks, IT admin expertise
• Spin up new machines in minutes
  • No costly purchase and setup of new hardware
• No need for local admin expertise
Virtualization

• A key part of cloud economics is **virtualization**
• **Virtualization** is the *emulation* of hardware resources to execute multiple independent operating systems under *one physical machine*
  • Basically, you can run a whole **operating system** as an “app”
Hypervisor-based virtualization
Hardware virtualization

• Gives guest OS the illusion that it has its own dedicated hardware
• Illusion may be stronger (VMWare hypervisor) or weaker (Xen)
  • Recent trend: containerization (docker)

• Provides relatively strong resource and security isolation between operating systems
• Great properties for shared hosting environments!
  • One giant physical server with lots of small virtual machines
  • Remember your AWS instance?
    • The free tier is ~512MB RAM. Who has that little RAM anymore?