6.033 in the news

“Unfortunately, the system did not scale fast enough to accommodate the increased volume”

“One such technology that the state could use is a cloud computing service that scales up server power to meet demand. Olivia Adams, a software developer who created MACovidVaccines.com while on maternity leave, uses one such service from Amazon. That’s why her website didn’t crash when the state’s did, she thinks.”

questions that arise:

Why didn’t the state’s website scale to meet demand? How do we test systems for that? How do we understand how far a system can scale, and in what dimensions?

Are companies like Amazon and Google the only places to get such highly-scaling services? If so, why? What implications does that have?

In particular, what are the consequences of public infrastructure relying on private companies?
6.033 Spring 2021

Lecture #2: Naming
plus a case-study on DNS
**last time:** enforced modularity via client/server

**today:** naming, which allows modules to interact
why use names?

they let us achieve modularity by providing communication and organization, as well as a number of other properties

**client**

```
Class Browser
(on machine 1)
```

```
def main():
    html = browser_load_url(URL)
    ...
```

```
def browser_load_url(url):
    msg = url  # could reformat
    send request
    wait for reply
    html = reply  # could reformat
    return html
```

**server**

```
Class Server
(on machine 2)
```

```
def server_load_url():
    ...
    return html
```

```
def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```

retrieval
the client can retrieve the llama page because it can name it

sharing
the server can share the llama page with multiple clients (i.e., multiple clients can view this page)

user-friendly IDs
kaws.com is easier to remember than (say) 18.25.4.171; the variable name “html” is easier to remember than a particular location in memory

addressing
some names also specify location information
**why use names?**

they let us achieve modularity by providing communication and organization, as well as a number of other properties.

**hiding**

code on the server can access `llama_data.txt` without having to worry about how the file is laid out in memory.

**indirection**

the server can change the memory layout of `llama_data.txt` without notifying the user.
why use names? they let us achieve modularity by providing communication and organization, as well as a number of other properties

the design of a system’s naming scheme(s) helps it achieve these properties

a naming scheme includes

1. the set of all possible names
2. the set of all possible values
3. a look-up algorithm to translate a name into a value (or a set of values, or “none”)
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to IP addresses (18.62.1.6)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a **zone**. name servers within a zone keep track of that zone’s mappings
naming case study: the **domain name system (DNS)**, which maps **hostnames** (eecs.mit.edu) to **IP addresses** (18.62.1.6)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

---

**A partial view of the DNS hierarchy.** Each box represents a **zone**. Name servers within a zone keep track of that zone’s mappings.
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.62.1.6)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation.

A partial view of the DNS hierarchy. Each box represents a zone. Name servers within a zone keep track of that zone’s mappings.

DNS client

query sent to: 192.14.171.191
try 18.72.0.3

domain name system (DNS)
naming case study: the domain name system (DNS), which maps hostnames (eecs.mit.edu) to IP addresses (18.62.1.6)

the look-up algorithm has to scale to the size of the Internet, while dealing with constant updates and issues of delegation

a partial view of the DNS hierarchy. each box represents a zone. name servers within a zone keep track of that zone’s mappings

DNS client e.g., your laptop query sent to: 18.72.0.3 response: 18.62.1.6
naming case study: the **domain name system (DNS)**, which maps hostnames (eecs.mit.edu) to **IP addresses** (18.62.1.6)

the **look-up algorithm** has to scale to the size of the Internet, while dealing with constant updates and issues of delegation.

---

**performance issue**: this is a *lot* of queries, especially to the root server.

**reliability issue**: what happens when a nameserver fails or (**security issue**) is attacked?

**control issue**: who should own the root server?

---

A partial view of the DNS hierarchy. Each box represents a **zone**. Name servers within a zone keep track of that zone’s mappings.
modularity and abstraction mitigate complexity. a client/server model allow us to enforce modularity by putting modules on physically separate machines.

naming is what allows modules to interact, and can help us achieve other goals through properties such as indirection, user-friendliness, etc.

the domain name system is a great case-study in naming, and also illustrates principles such as hierarchy, scalability, delegation, and decentralization

and client/server models, and (tomorrow) caching, and (in May) security…

the example you saw in lecture was a fairly basic one; you will talk more about DNS’s performance enhancements in recitation tomorrow, which change how some (many) DNS queries are resolved