Recitation 10

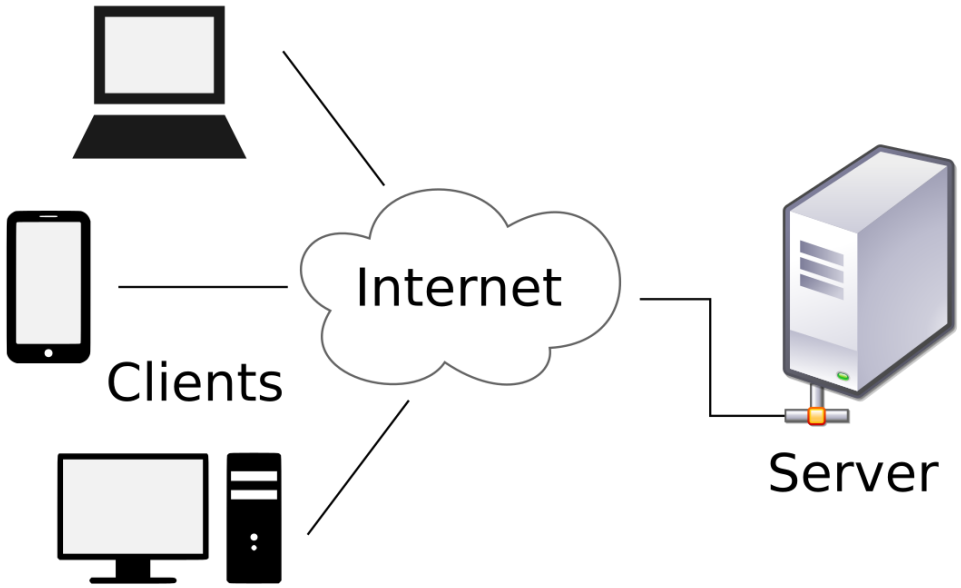
Yifan Wang

Lambda Model
Serverless Computation

What is Serverless Computation?

How do we provide Internet service?

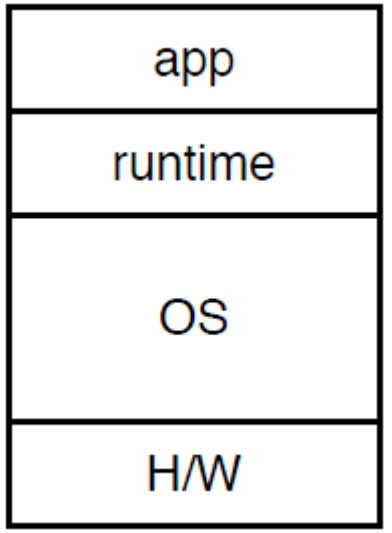
Client-Server Model



https://en.wikipedia.org/wiki/Client%E2%80%93server_model

- Servers

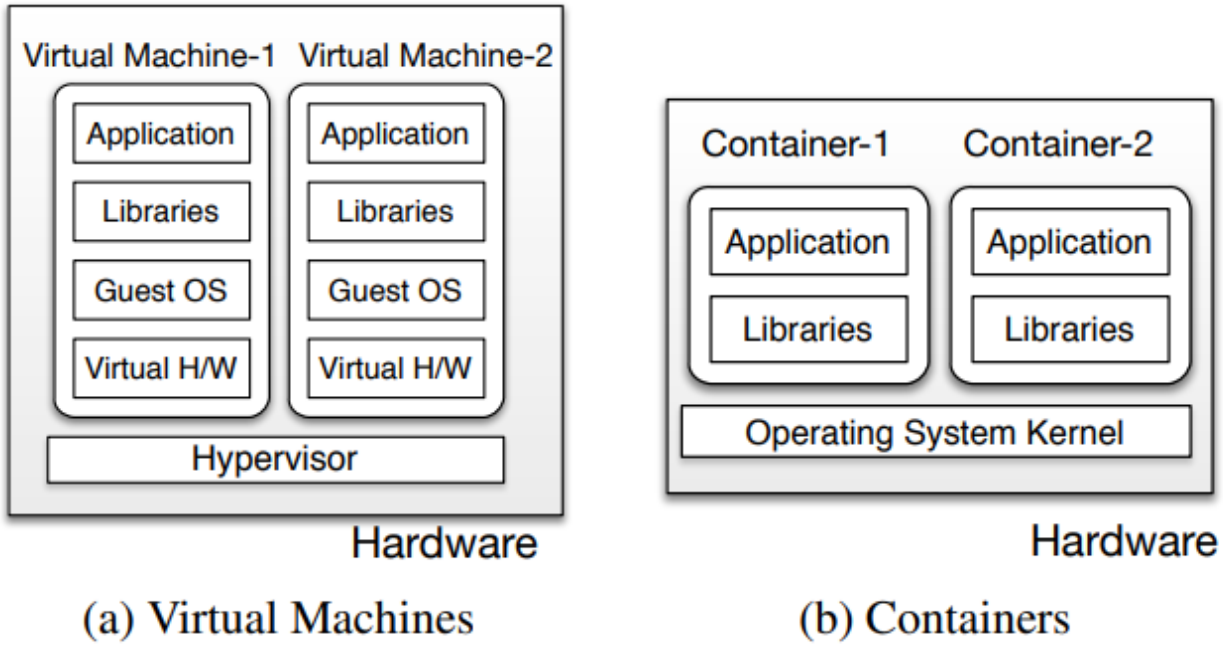
https://www.google.com/about/datacenters/gallery/



Virtualize a SERVER.

a) Virtual Machines
   - Hardware-level virtualization.
   - Hypervisor emulates hardware.
   - VMware ESXi, Linux KVM.

b) Containers
   - OS level virtualization.
   - Namespace isolation.
   - Docker, LXC

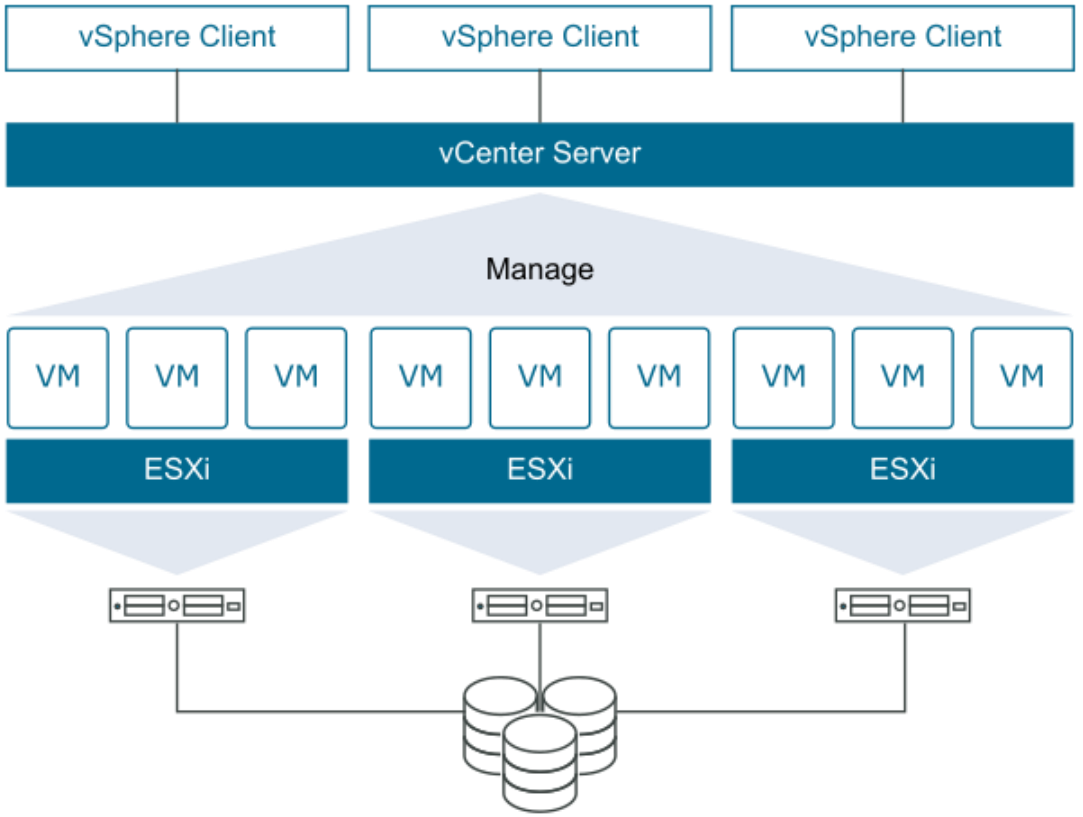


(a) Virtual Machines (b) Containers

https://people.cs.umass.edu/~prateeks/papers/mw_submitted.pdf

Virtualize servers, aka. DATACENTER.

- Single server management:
  - Deploy VMs.
  - Power cycle.
  - Hardware configuration, e.g., NIC, GPU, etc.
- Cross server management:
  - VM migration.
  - Cluster/Datacenter wide resource allocation.
- Software defined network:
  - Management vs. data.
  - Switches.
- Software defined storage:
  - Datacenter wide memory and disk management.
- Performance specification oriented.



https://docs.vmware.com/en/VMware-vSphere/index.html

Virtualization from coder's view

- Gray areas:
  - Provided by cloud vendors.
  - Shared by multiple users.
- Before:
  - Server-centric.
  - "Servers are notoriously difficult to configure and manage"
  - "startup time severely limits an application's ability"
- After:
  - Data centric.
  - Serverless computation.
  - Lambda model.

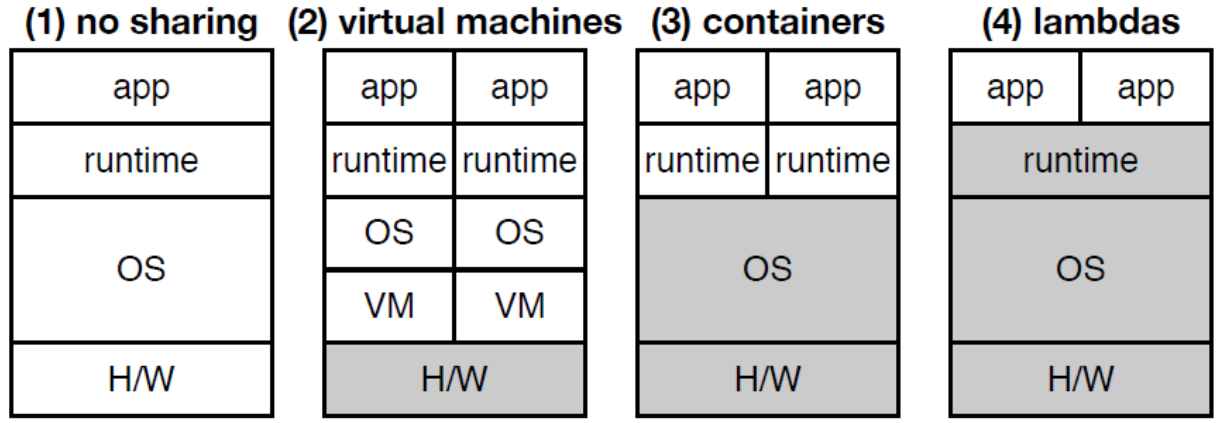


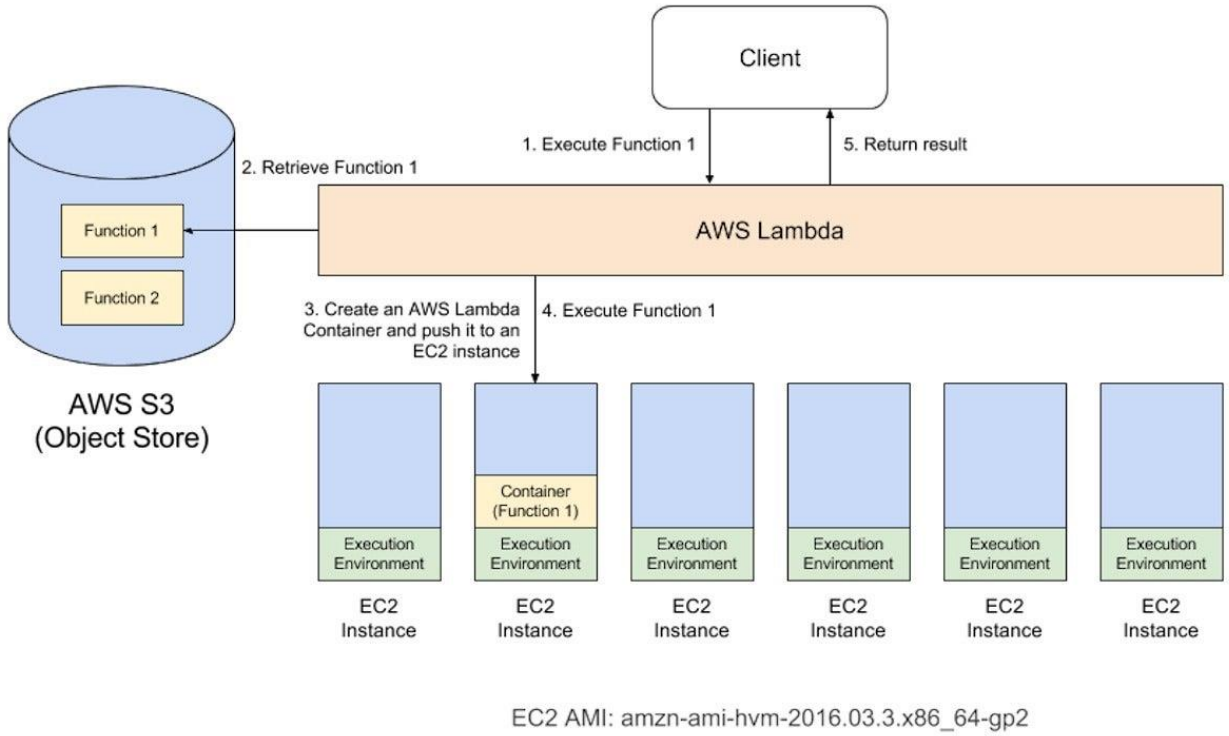
Figure 1: **Evolution of Sharing.** *Gray layers are shared.*

What is the Lambda model?

- Simple idea:
  - Programmer provides callback functions.
  - Call back functions are triggered by events.
- Widely adopted by cloud vendors:
  - Amazon Lambda.
  - Azure Serverless.
  - Google cloud serverless computing.

How is Lambda model implemented?

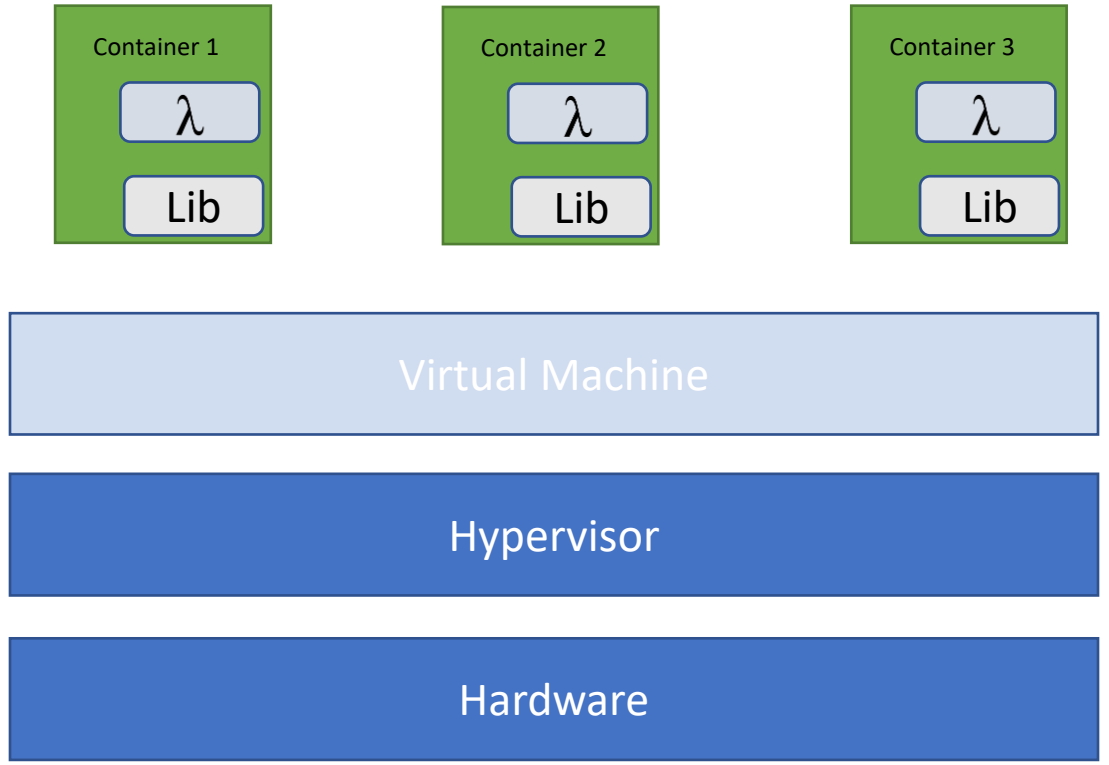
- User provides a .zip package of files or container image.
- AWS store the file in S3.
- At invocation, AWS creates a container on its “farm” of EC2 instances.
- Run the container and execute the call-back functions.



https://www.infoworld.com/article/3210726/serverless-computing-with-aws-lambda.html#:~:text=AWS%20Lambda%20encrypts%20and%20stores,farm%2C%20and%20executes%20that%20function.

Serverless computation from Amazon' view

- Hypervisor is still needed to manage servers.
  - Better isolation
  - Easier control at datacenter level.



Is Amazon Lambda expensive?

- Charge by duration and request
  - Lambda counts a request each time it starts executing in response to an event notification trigger
  - Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 1 ms

AWS Lambda Pricing

Region: US West (Northern California)

Architecture	Duration	Requests
x86 Price		
First 6 Billion GB-seconds / month	$0.0000166667 for every GB-second	$0.20 per 1M requests
Next 9 Billion GB-seconds / month	$0.000015 for every GB-second	$0.20 per 1M requests
Over 15 Billion GB-seconds / month	$0.0000133334 for every GB-second	$0.20 per 1M requests
Arm Price		
First 7.5 Billion GB-seconds / month	$0.0000133334 for every GB-second	$0.20 per 1M requests
Next 11.25 Billion GB-seconds / month	$0.0000120001 for every GB-second	$0.20 per 1M requests
Over 18.75 Billion GB-seconds / month	$0.0000106667 for every GB-second	$0.20 per 1M requests

Is Amazon Lambda expensive?

Service	Price
Provisioned Concurrency Pricing	0.0000049041 for every GB-second.
Data transfer	Calculated in S3 and other DB services.
Lambda@Edge Pricing	0.60 per 1M requests 0.00005001 for every GB-second

How does Amazon Lambda perform?

- Comparison group:
  - AWS Elastic Beanstalk.
  - Container-based solution.
  - Emphasizes easy deployment.
- Benchmark:
  - Maintains 100 outstanding RPC requests and each RPC handler spins for 200ms.
  - Believed to be overwhelming for Elastic BS.

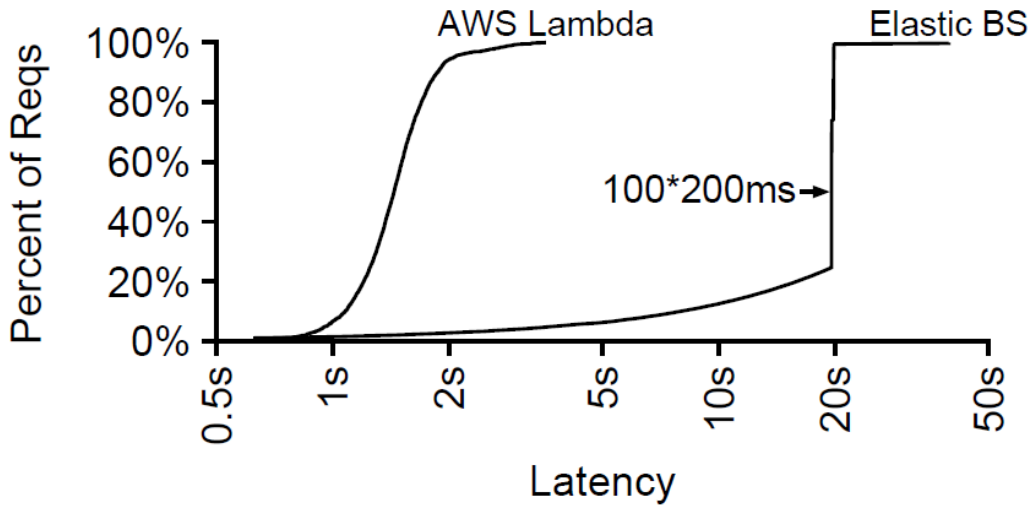


Figure 2: **Response Time.** *This CDF shows measured response times from a simulated load burst to an Elastic BS application and to an AWS Lambda application.*

How does Amazon Lambda perform, cont'd?

	Lambda	Elastic BS
Median	1.6s	20s
Instances	100	1

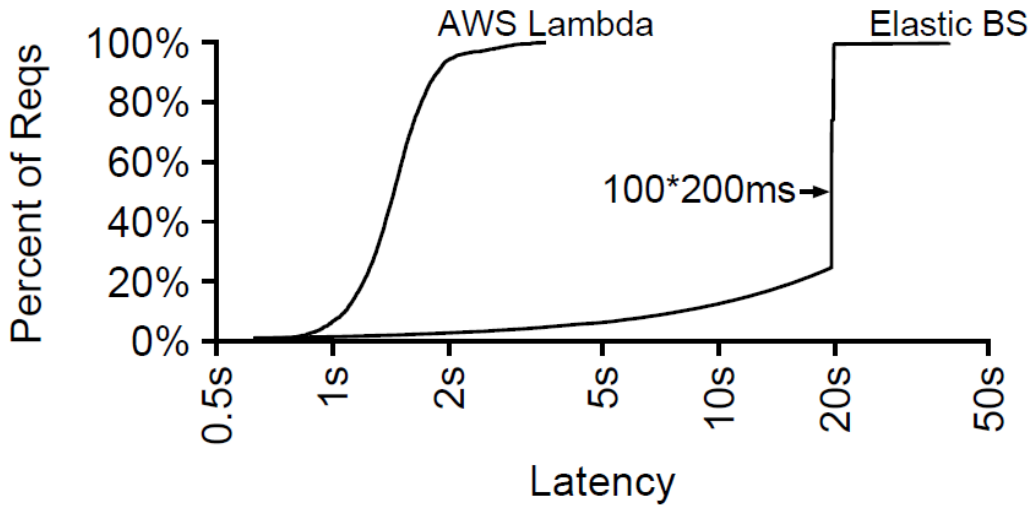


Figure 2: **Response Time.** *This CDF shows measured response times from a simulated load burst to an Elastic BS application and to an AWS Lambda application.*

Container startup takes a long time.

Low workload Elastic BS vs. Lambda.

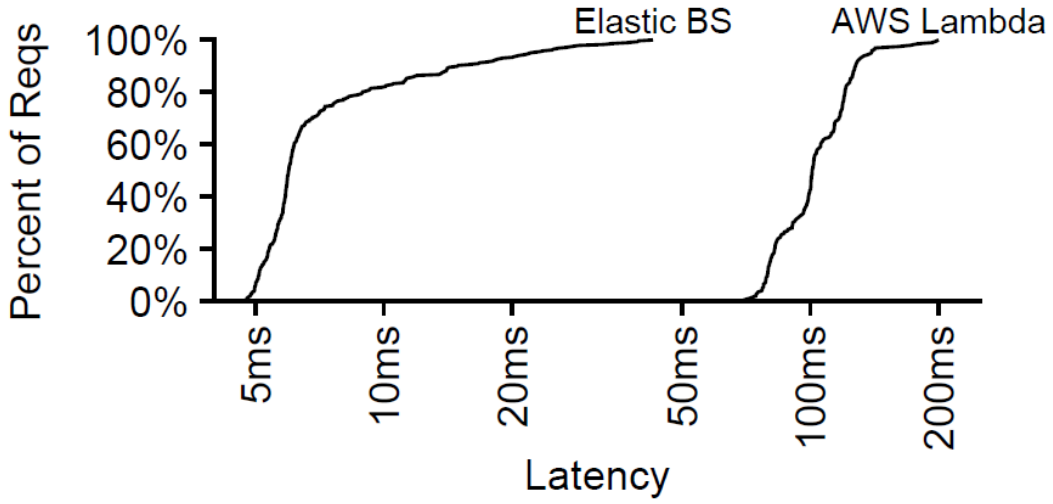


Figure 4: **Containers vs. Lambdas: Latency.** *The lines represent a latency CDF for AWS Lambdas, and containers running in Elastic BS, with both services under low load.*

Container startup time.

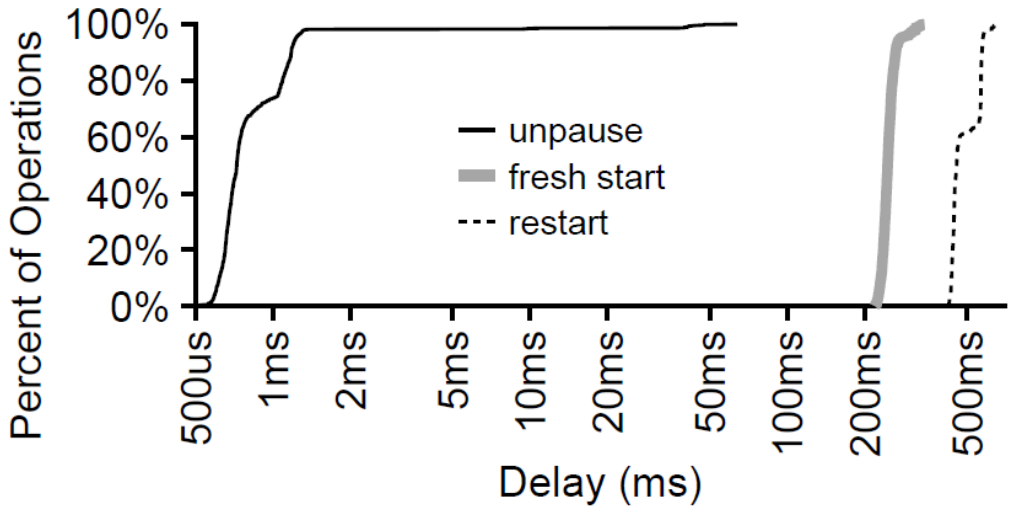


Figure 5: **Readiness Latency.** *The lines represent readiness latency CDFs for three startup transitions.*

It can be expensive to keep all containers live.

Memory consumption.

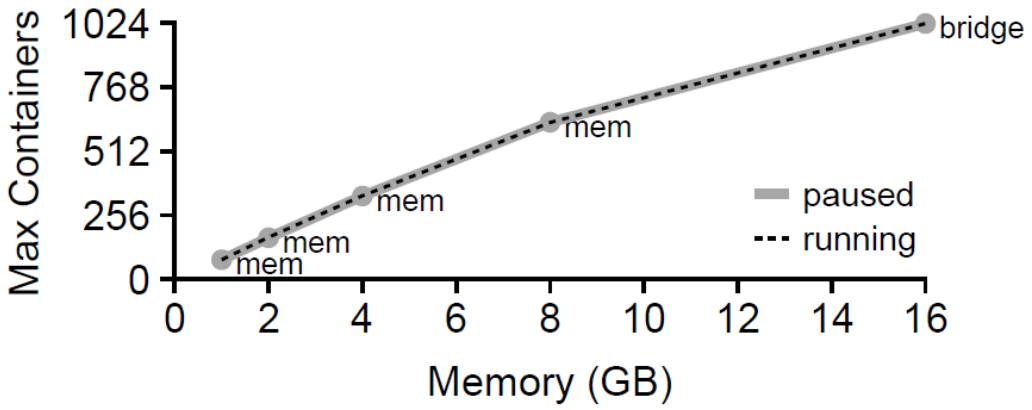


Figure 6: **Container Density.** *The maximum number of containers a machine can run is shown vs. the amount of memory available. Labels identify bottlenecks.*

Provisioned concurrency charging.

Region: US West (Northern California)

Architecture	Provisioned Concurrency	Duration	Requests
x86 Price	$0.0000049041 for every GB-second	$0.000011443 for every GB-second	$0.20 per 1M requests
Arm Price	$0.0000039233 for every GB-second	$0.0000091544 for every GB-second	$0.20 per 1M requests

Can we use Lambda Model for real web apps?

How do web apps “talk”?

How do web apps “talk”, cont’d?

- AJAX
  - **A**synchronous **J**avaScript **A**nd **X** ML.
  - Use XMLHttpRequest object to communicate with servers.
  - Use JavaScript to manipulate DOM.
  - Built on HTTP request:
    - POST and GET are the 2 verbs used.
    - Synchronized and asynchronized modes are supported.
    - Easy to use.

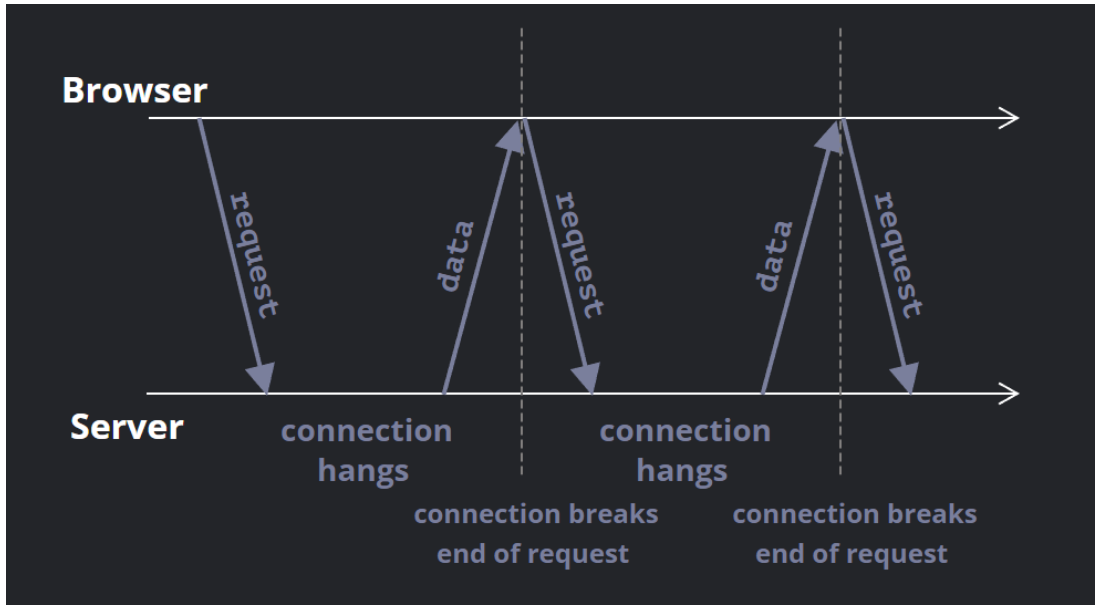
```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
    }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

https://www.w3schools.com/JS/js_ajax_intro.asp

Can we use AJAX for WebSocket-flavor tasks?

Long polling:

1. A request is sent to the server.
2. The server doesn't close the connection until it has a message to send.
3. When a message appears – the server responds to the request with it.
4. The browser makes a new request immediately.



https://javascript.info/long-polling#:~:text=Long%20polling%20is%20the%20simplest,in%20a%20lot%20of%20cases.

Profile GMail

- How:

  Create a chrome extension (client side) to inject a wrapper to measure network flow.

- POST and GET:
  - POST for RPC, GET for others
  - RPC: 92ms vs. Others: 18ms
- RPCs:
  - Short RPCs (<100ms) has median of 27ms
  - Potential long polling RPC exists.

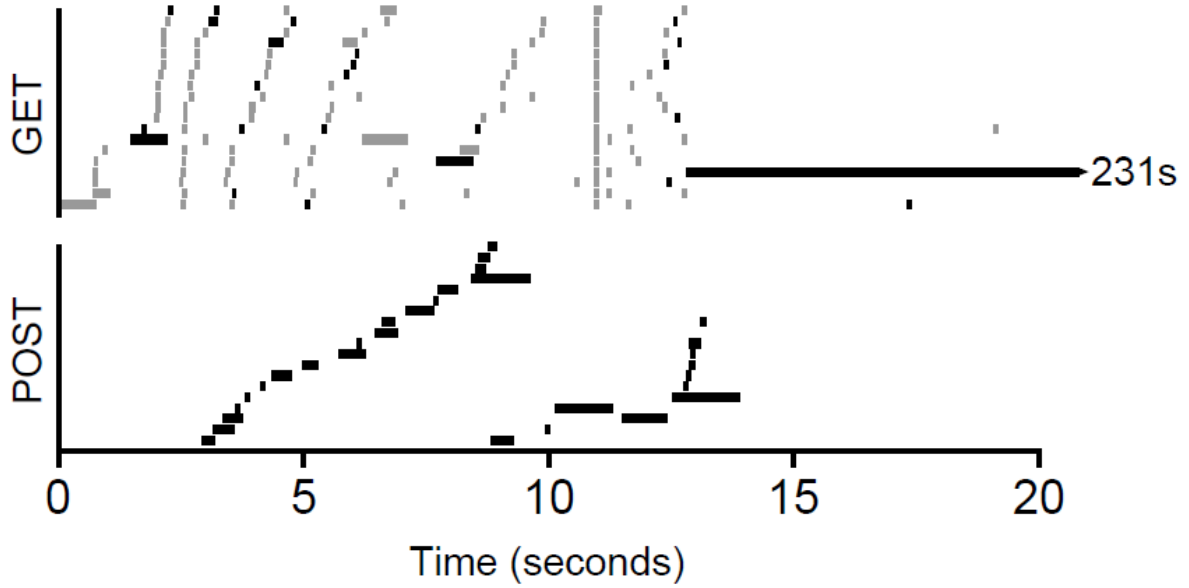


Figure 3: **Google Gmail.** *Black bars represent RPC messages; gray bars represent other messages. The bar ends represent request and response times. The bars are grouped as POSTs and GETs; vertical positioning is otherwise arbitrary.*

Map Reduce

We can use a hierarchy of lambda functions to perform some complex tasks.

Long Long Ago… Lisp 1958

Map

```
(defun squareMylist(lst)
   (map 'list (lambda(x) (* x x )) lst)
)
(write(squareMylist '(2 3 4 5 6 7 8 9)))
```

```
$clisp main.lisp
(4 9 16 25 36 49 64 81)
```

Reduce

```
(defun squareMylist(lst)
   (map 'list (lambda(x) (* x x )) lst)
)
(setq intermediateResult (squareMylist '(2 3 4 5 6 7 8 9)))
(write(reduce #'+ intermediateResult))
```

```
$clisp main.lisp
284
```

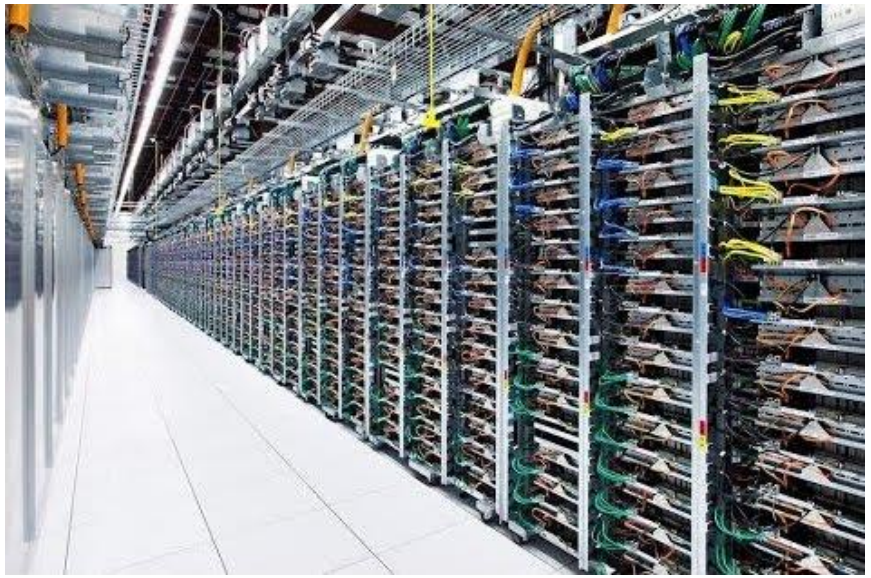
Google File System 2003

What Does Google Have?

- Computer Farm:
  - Huge Amount
  - Inexpensive computers
  - Local Disk &NICs
  - Unreliable hardware
- GFS (Ghemawat2003):
  - Distributed file system
  - Reliable service
  - Acceptable throughput and latency

What Does Google Want?

- Index webpages fast
  - Simple algorithm
  - Vast amount



Map-Reduce 2004

Map-Reduce Paradigm (Dean2004)

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

```
(defun squareMylist(lst)
   (map 'list (lambda(x) (* x x )) lst)
)
(write(squareMylist '(2 3 4 5 6 7 8 9)))
```

```
(write(reduce #'+ intermediateResult))
```

Map() and Reduce()

- Linear Regression

$$y = \theta^T x$$

- Mean Square Error

$$Error = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x_i - y_i)^2$$

- Optimization

$$\theta^* = min_\theta \sum_{i=1}^{m}(\theta^T x_i - y_i)^2$$

- Solution

$$\theta^* = (X^T X)^{-1} X^T Y$$

$$A = \sum_{i=1}^{m}(x_i x_i^T)$$

$$b = \sum_{i=1}^{m} x_i y_i$$

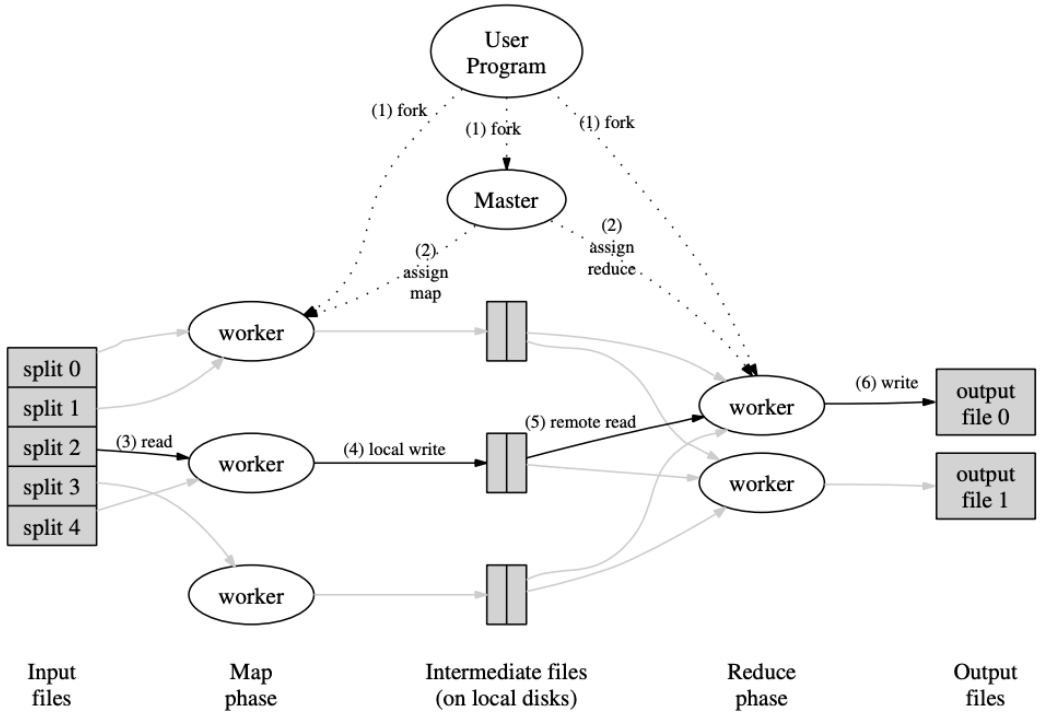
Map() and Reduce(), cont'd

Map()

Component	Input	Output
A	X: [x_0, x_1, x_2, …]	$[x_i x_i^T]$
b	X: [x_0, x_1, x_2, …] y: [y_0, y_1, y_2, …]	$[x_i y_i]$

Reduce()

$\sum$

Google's Solution

- Split input data to 64MB blocks.
- Store data in local file system.
- Assign mappers to process its local copy.
- Mapper write intermediate result to local disk.
- Assign reducer to aggregate result.
- Write output to local file system.