

Deploying Large Scale Webapps

Thierry Sans

Users respond to speed

“Amazon found every 100ms of latency cost them 1% in sales”

“Google found an extra .5 seconds in search page generation time dropped traffic by 20%”

“A broker could lose \$4 million in revenues per millisecond if their electronic trading platform is 5 milliseconds behind the competition”

<http://blog.gigaspace.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/>

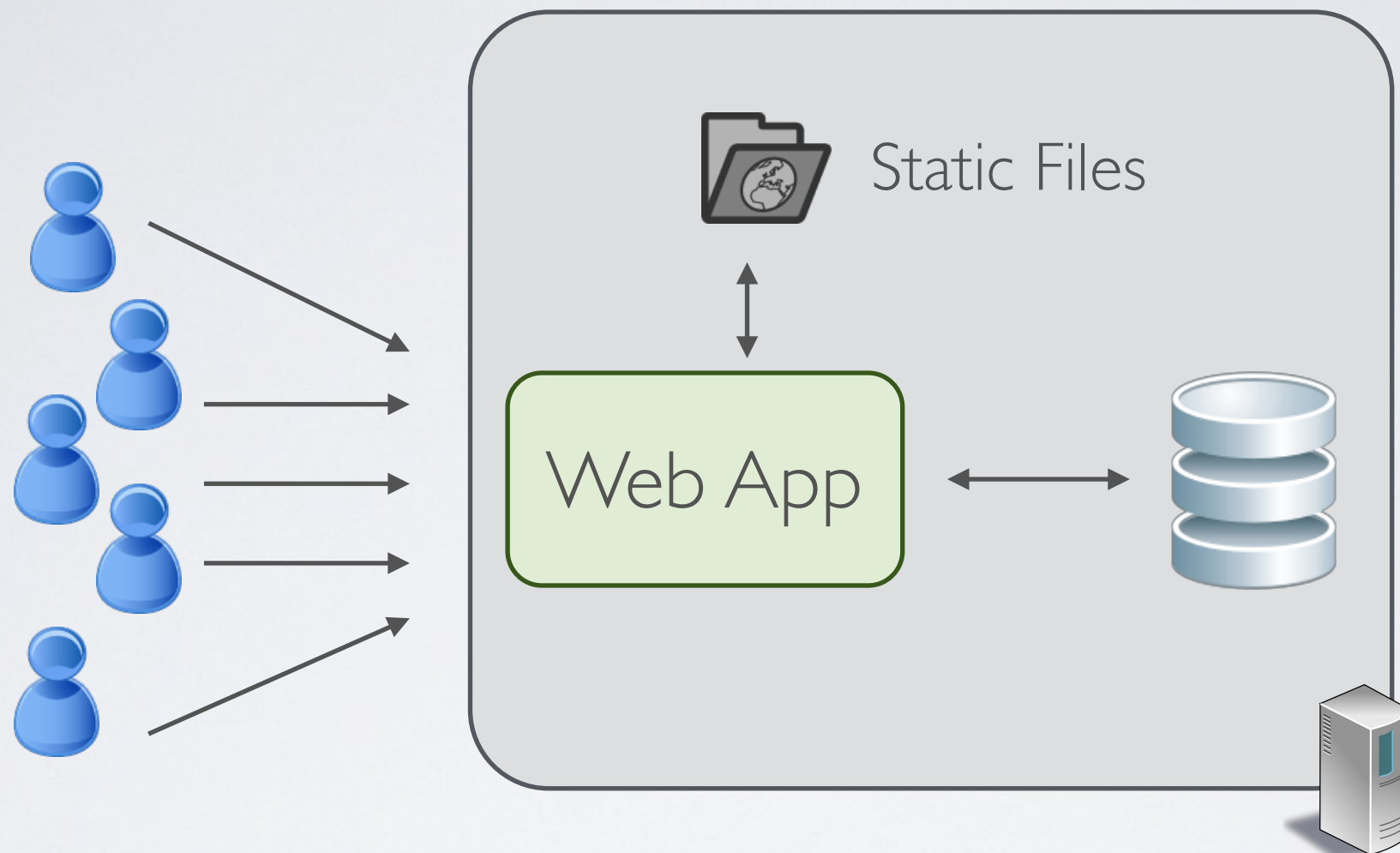
How to serve millions

Optimizing frontend code with HTTP/2, Web Workers and PWA (Progressive Web Apps)

Optimizing backend code with web caching

Optimizing the infrastructure with microservices

Current situation (dedicated or virtual server)



Two types of content

- Static content : html, css, js, images and so on
- Dynamic content : database, uploaded files

Optimizing the backend code with Web Caching

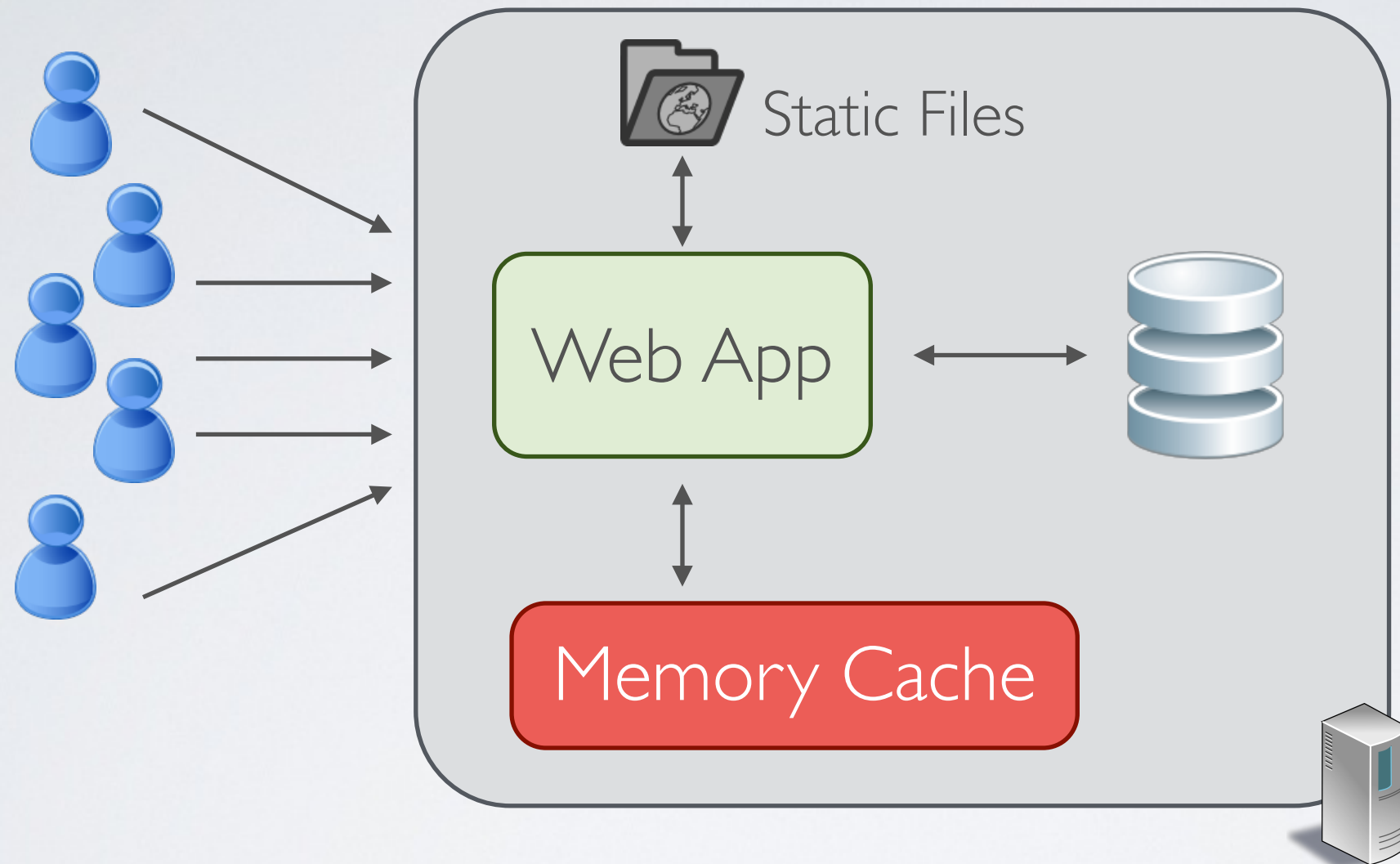
How to improve response time?

Processing the request means:

1. Parse the HTTP request
2. Map the URL to the handler
3. Query the database or third-party API
4. Compute the view

DB and API accesses are expensive (time and money when your host charges you each access)

Fine-grained caching with the web application (for dynamic content)



Cache controlled by the program

- Specific for each app
- ✓ Good for dynamic content
- ➔ Popular memory cache: Memcached

Distributed Shared Cache : Memcached

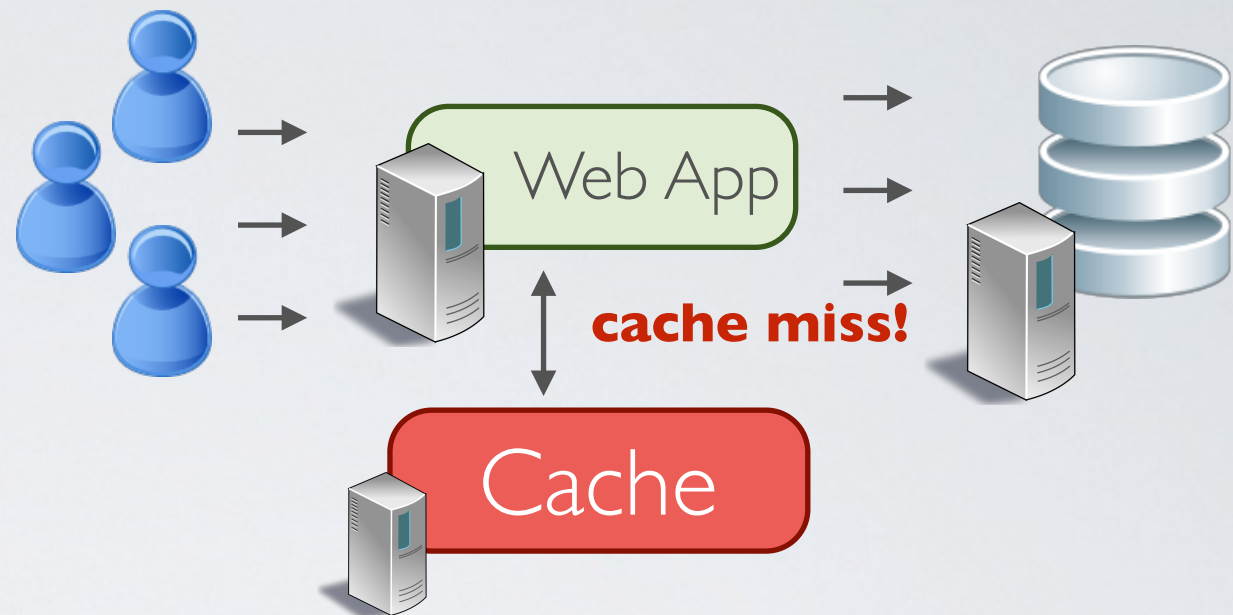
<http://memcached.org/>

- Store key/value pairs in memory
- Throw away data that is the least recently used

A typical cache algorithm

```
retrieve from cache
if data not in cache:
    # cache miss
    query the database or API
    update the cache
return result
```

Cache Stampede (a.k.a dog piling)



Problem:

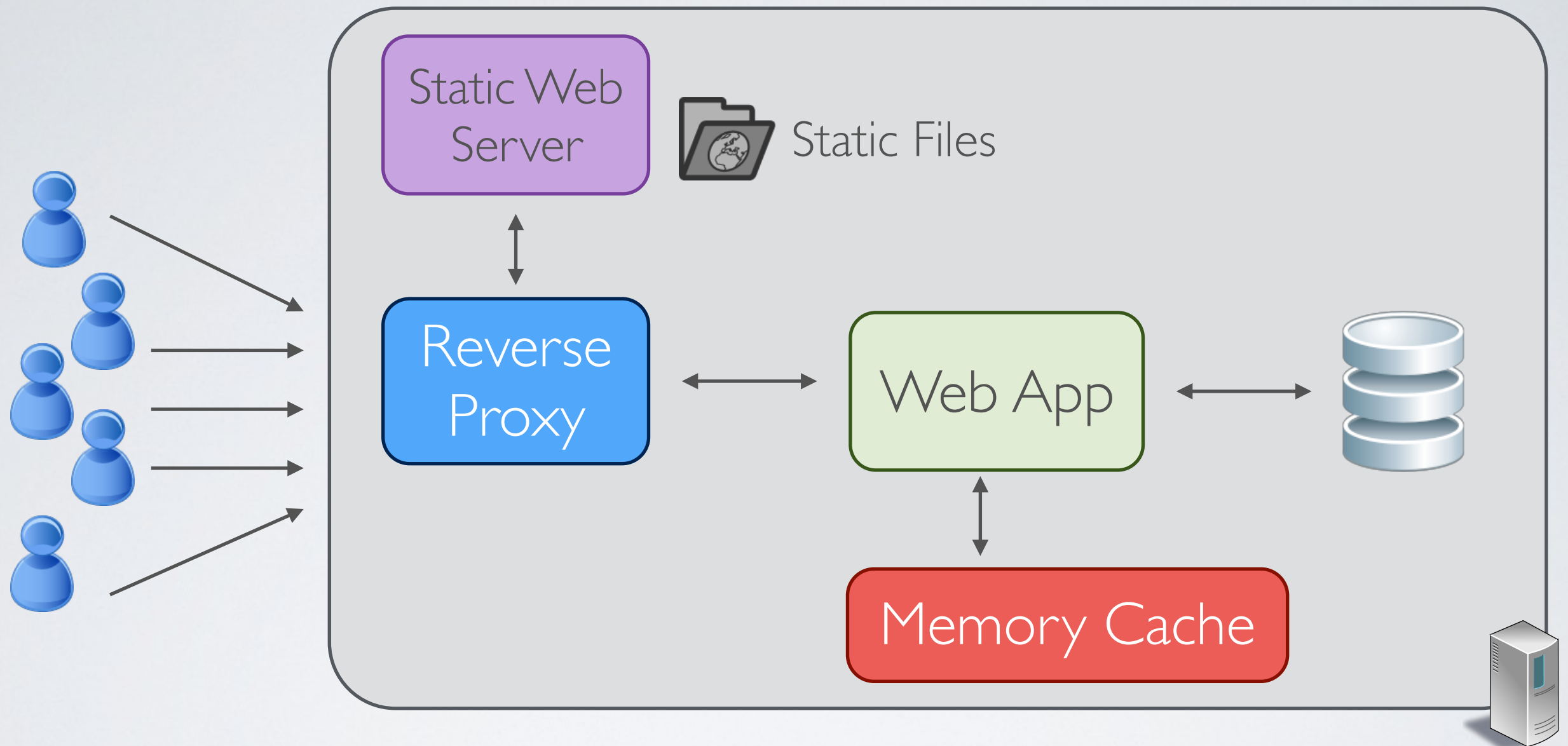
Multiple concurrent requests doing the same request because cache was cleared

Solution:

- update the cache instead of clearing it after an insert
 - a page view will never query the database
- ➡ Requires cache warming

Optimizing the infrastructure with microservices

Optimized Static Content

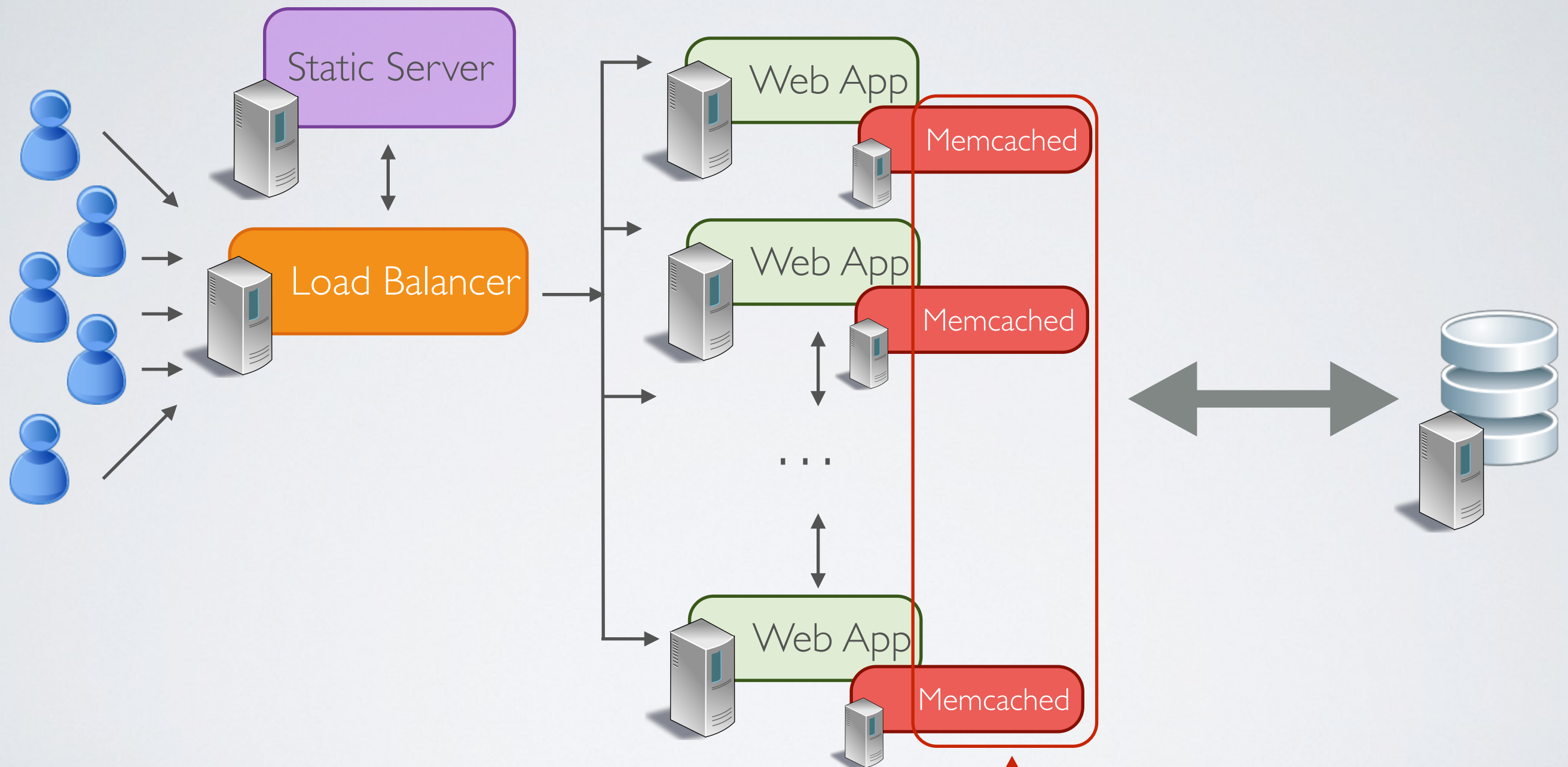


Cache repeated HTTP requests for a given time

- Bad for dynamic content (latency when the content is updated)
- ✓ Good for static content (Javascript, CSS, Media, static HTML)

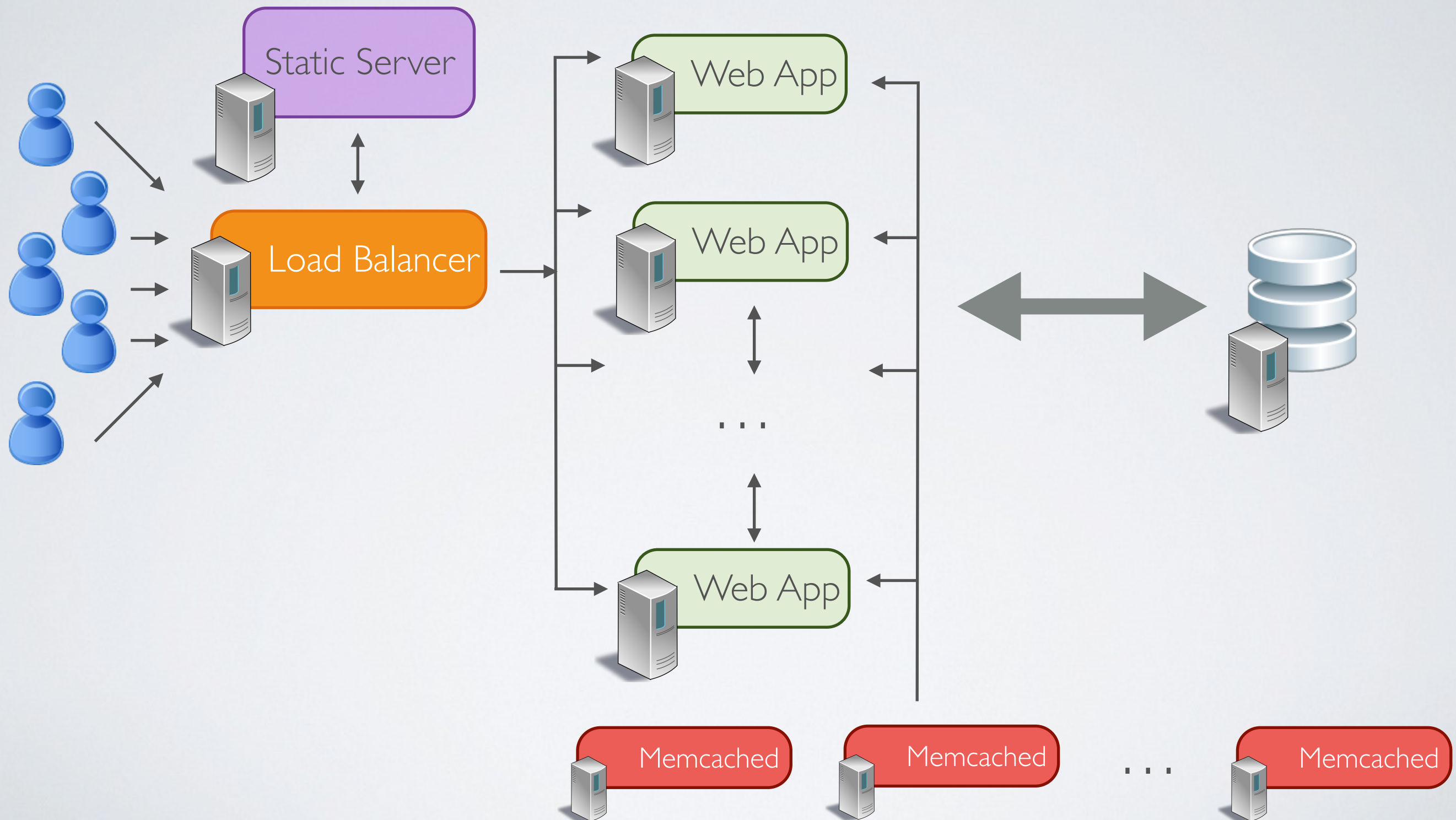
Scaling over multiple servers

Serving multiple apps with a load balancer

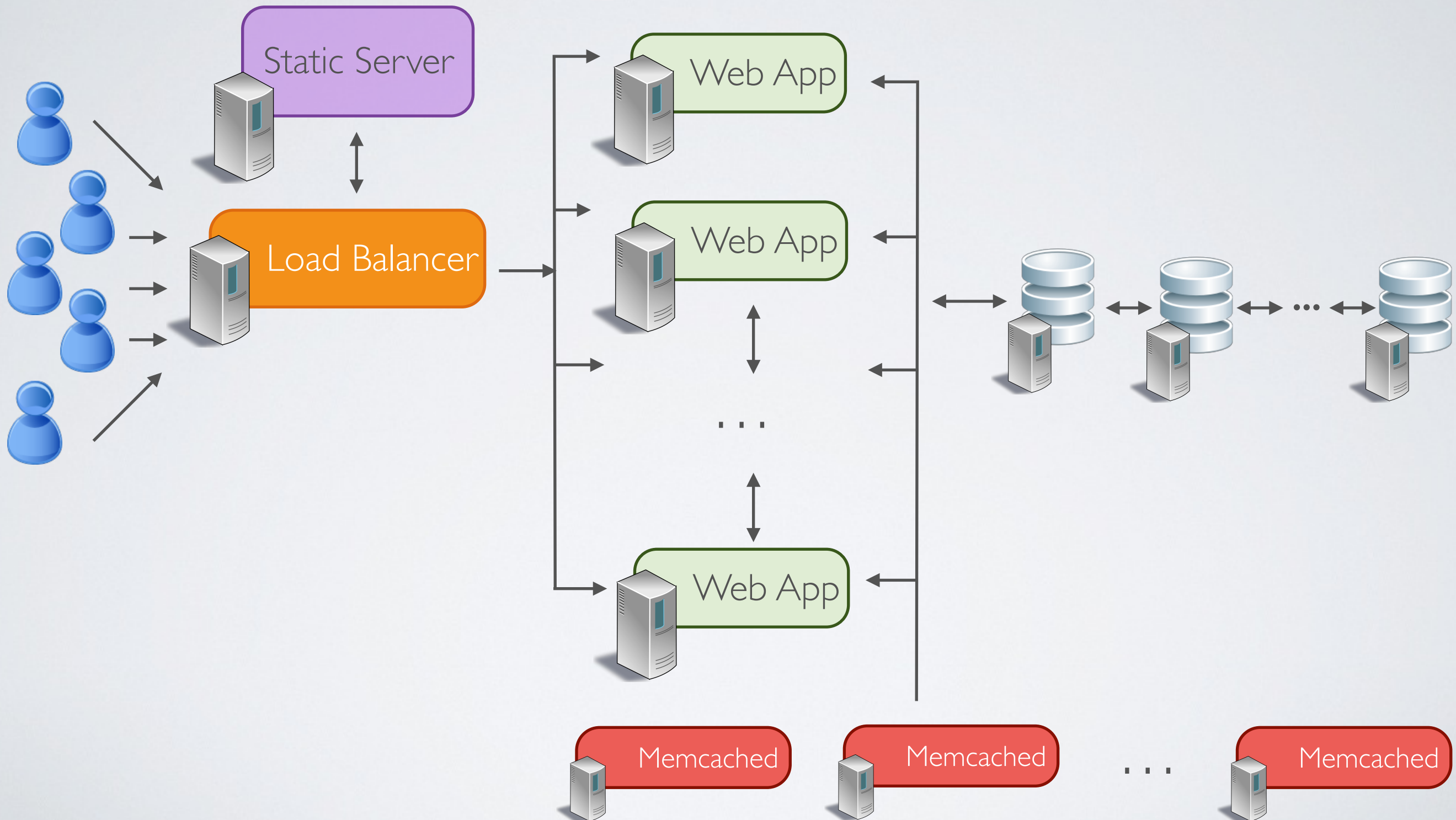


This is not an efficient cache

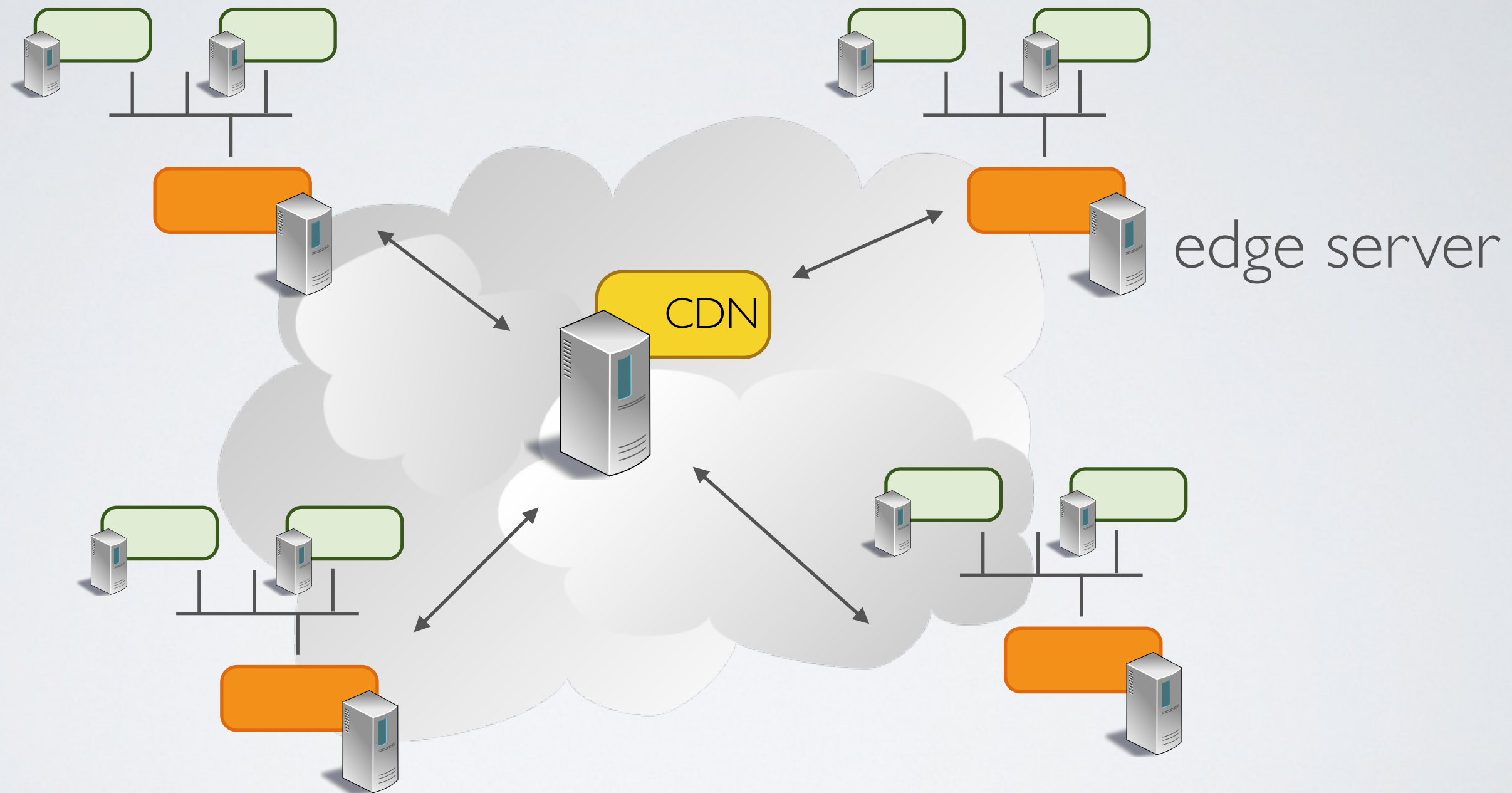
Distributed Shared Cache



Distributed Databases



CDN : Content Distribution Network



Example : Akamai, Cloudflare

High-Performance Software

Reverse proxy	Nginx
Static Web Server	
Load Balancer	
Memory Cache	Memcached / Redis
Container	Docker
Container Orchestration	Kubernetes / Docker Swarm