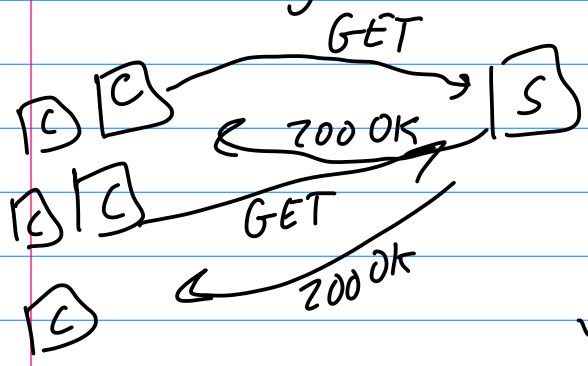
CS 3214 lecture #26 "web servers: threads vs. events"

HTTP + cloud computing

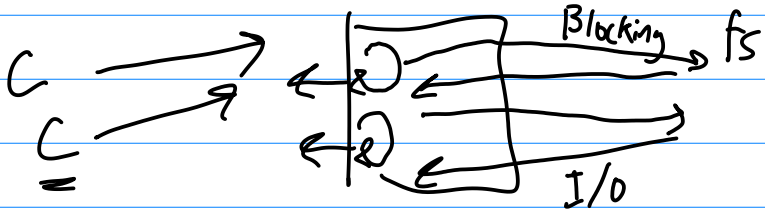
SPOT survey - 14%

Concurrency -

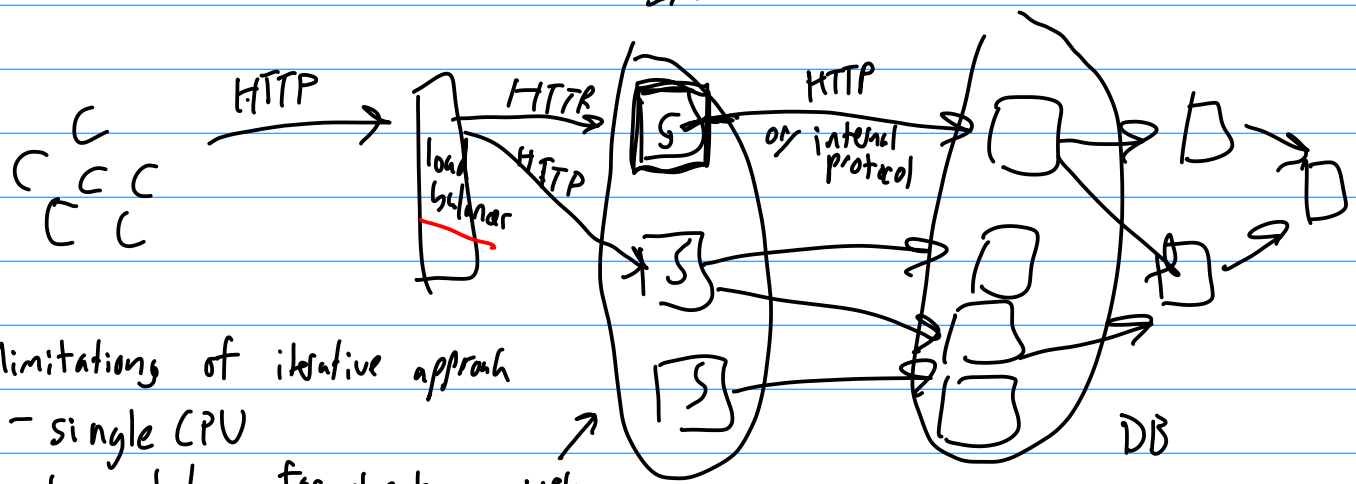


accepts client connection
→ processing
sends response

"iterative server"
- one req at a time (sequentially) serially



low utilization



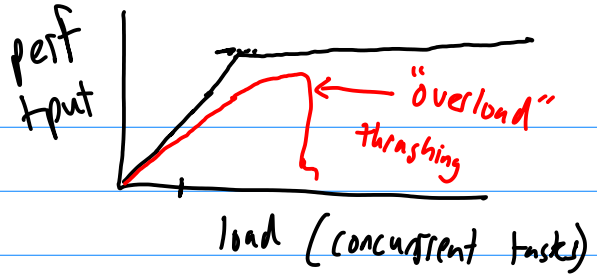
limitations of iterative approach
- single CPU
- high latency for client
→ - low utilization

web servers

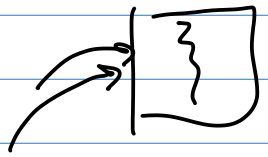
back-end services

"Microservices"
"service mesh"

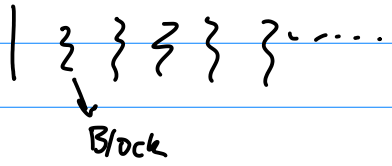
handle 1978 multiple requests
- { threads / processes — concurrently (apache)
- { event-based (nginx)



Thread-based approach
- underlying system to maintain context + state



PROS
- familiar linear control flow (blocking is fine)
- UNIX makes it easy
- isolation (process-based)



pool

fixed amount of thread
how to pick n?

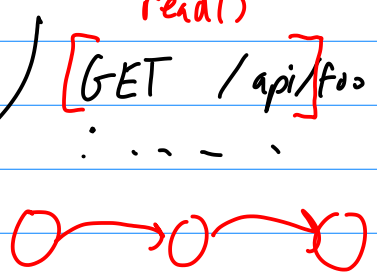
CONS
- lots of clients = LOTs of threads
  - state transitions, ctxt switches, mode switch
- concurrent programming is hard
- tuning (# threads

Event-based model
reorganize program into event handlers
application manage client states
application decides what runs when
Nothing blocks

→ how to avoid blocking?
→ how to mange state

How to write event-based program
- identify where progam would block
- where do events come from
- maintain client state

parsing HTTP header.
read()



How to identify events?
- new request arrived
- new data recv'd
- connection closed
...

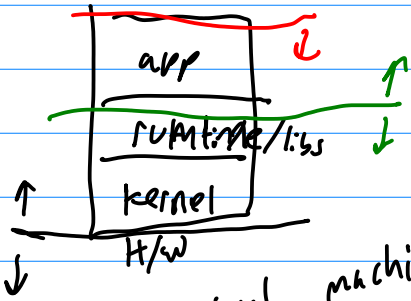
fd's ← which fd's have data pending?
or: will a read block on this fd?
will a write block on this fd?

- Non-blocking mode -EWOULDBLOCK
- select ← tell me when fd's are ready
poll
epoll ← fastest, most features (on Linux)

libraries: libevent, libuv, put abstraction layer
↑ node.js

High-level language support: async/await in Javascript
← make it look a bit more thread-like

+ speed,
+ no need to tune # threads
- not easy to program



Cloud computing ←
X as service
ex: EC2
infra - user s/w kernel, runtime, apps / cloud manages power, cooling etc.
platform - user s/w above a runtime ex: Heroku
software - entire application in cloud

IaaS PaaS SaaS

deployment - : cloud
: hybrid cloud
: on-prem

defining characteristics
- elasticity: can scale up & down instances on cloud (sometimes autoscale)
only pay for what you use
granularity ↓
- multi-tenancy
- isolation

[[cloud, VMs, containers]]