

Think before you develop!

Kai 'Oswald' Seidler  
Sun Microsystems

Codebits 2009, Lisbon

Kai 'Oswald' Seidler

***...APACHE  
FRIENDS...***

promote Apache web server



XAMPP

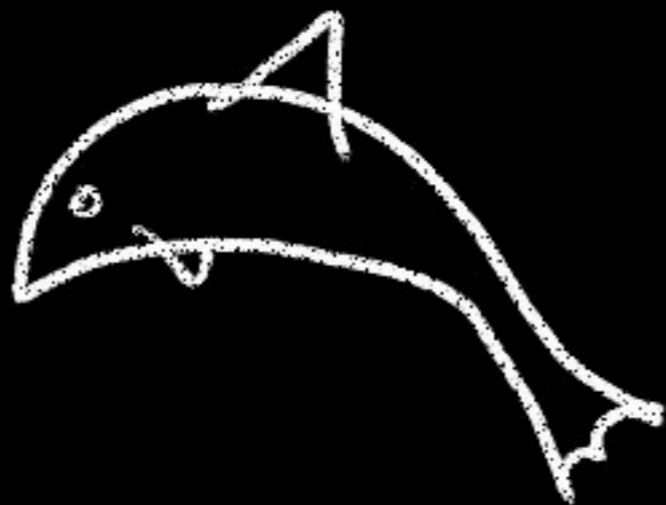
free

cross platform

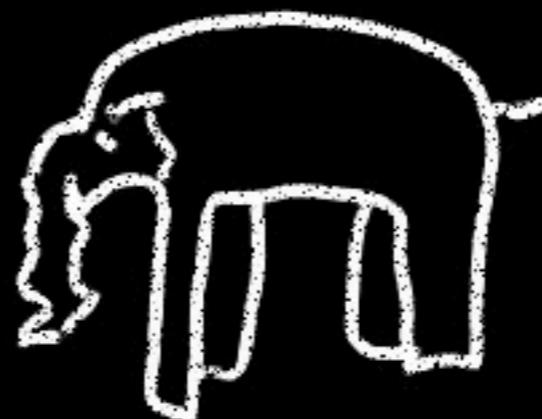
bundle



Apache



MySQL



PHP



Perl

very easy to use

unpack & start

beginners

world of web development

Windows

Linux

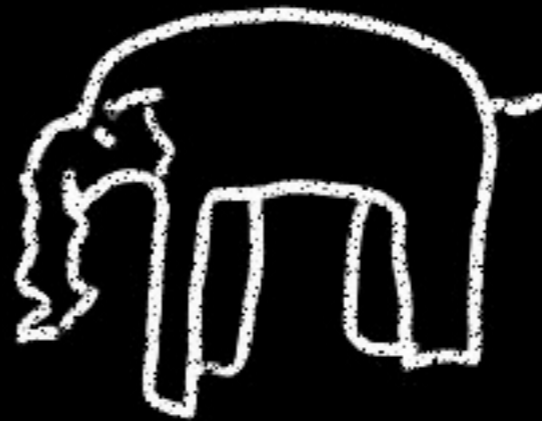
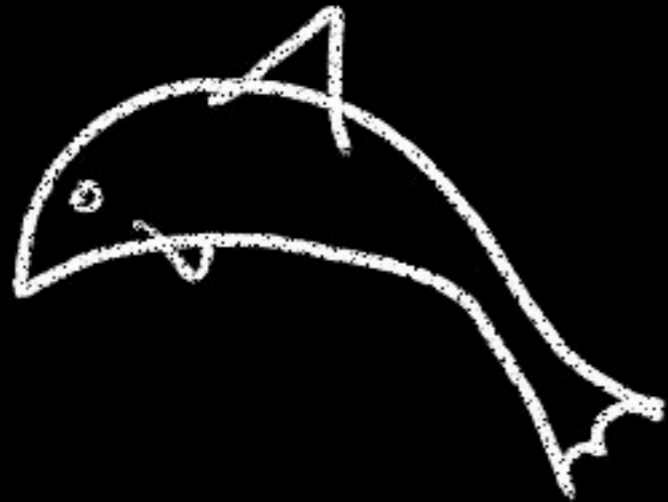
Mac OS X

Solaris





Sun GlassFish Web Stack



Apache

MySQL

PHP

Perl

GlassFish

Tomcat

Ruby

Python

Lighttpd

Squid

memcached

free

easy to use

production use

Solaris

OpenSolaris

RHEL

42 thoughts

web development

theoretical

practical

helpful

agenda

architectures

Languages

databases

gos & no-gos

scaling

<motivation>

why?

think!

no fun

hurts the brain

but help

avoid problems

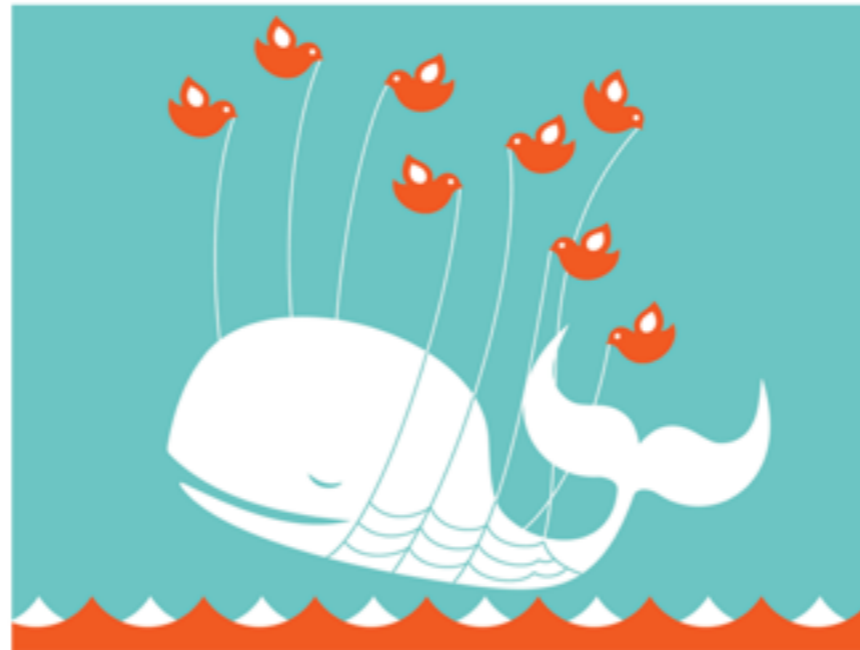
in the beginning

too late

two examples

**Twitter is over capacity.**

Too many tweets! Please wait a moment and try again.





We'll be  
back  
soon.

We are busy updating the store  
for you and will be back shortly.

You can contact our telesales team at the following numbers:

 US	1-800-MY-APPLE	 Nederland	0800 0200 570
 Australia	133-622	 New Zealand	0800-69-27753
 België	0800/99 846	 Norge	800-33-034
 Belgique	0800/93 932	 Österreich	0800 201 037
 Canada	1-800-MY-APPLE	 Philippines	0800 201 037
 Danmark	80-24-08-35	 Portugal	800 207 758

fixing this later



\$\$\$

Remove the cause  
but not the symptom.

Dr Frank. N. Furter

coding standards

naming conventions

documentation

development 4 production

let's start

<architectures>

4 types

software architecture

hardware architecture

transport architecture

service architecture

<1>

software architecture

multi-tier architecture

3 tiers

1.

presentation tier

aka "the user interface"

2.

application tier

aka "the software"

3.

data tier

aka "the database"

doesn't fit

for web development

client



server

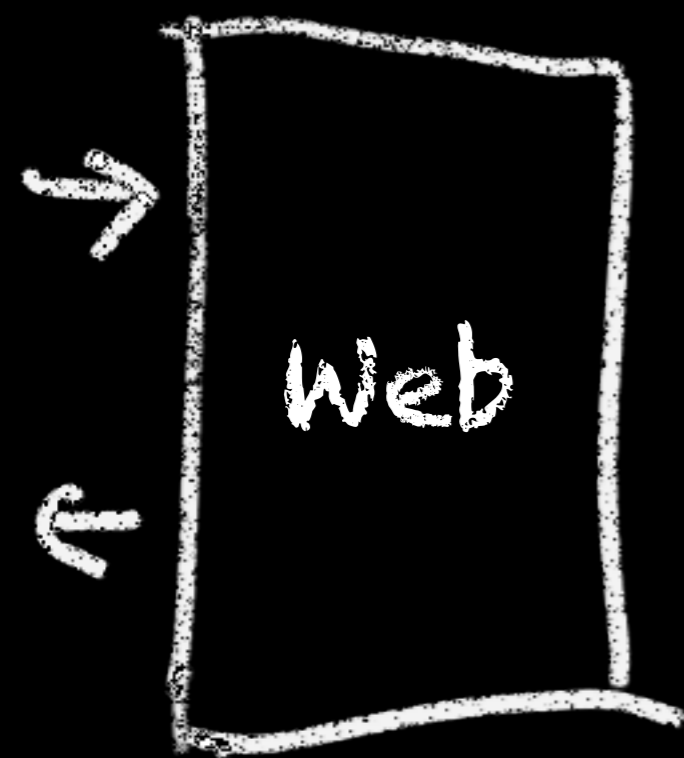


<2>

hardware architecture

web environment

3-tier architecture



static  
data



logic



variable  
data

<3>

transport architecture

server  $\leftrightarrow$  server

browser  $\leftrightarrow$  server

data-interchange formats

<4>

**XML**

server  $\leftrightarrow$  server ✓

Atom ✓

RSS ✓

browser <-> server ☢

to huge

to hard to parse

too much load 4 browser

<5>

JSON ✓

# JavaScript Object Notation

browser  $\leftrightarrow$  server ✓

server  $\leftrightarrow$  server ✓

very slim

easy to parse

service architecture

<6>

SOAP 

# Simple Object Access Protocol

XML-based format

very mighty

very huge

hard to work with

server  $\leftrightarrow$  server 🌧️🕒

browser <-> server ☢

<7>

REST ✓

REpresentational State Transfer

simple query language

based on HTTP

easy to implement

everyone speaks HTTP

server  $\leftrightarrow$  server ✓

browser <-> server ✓

JSON

Atom

<languages>

<8>

libraries

reinventing the wheel 🚫

reuse code

somebody else

has probably

written very well

CPAN

PEAR

Java APIs



<9>

frameworks

reuse code

reuse behavior

programming pattern

programming convention

software architecture

team work ✓

avoids spaghetti code

limits flexibility 🌂

Lower performance ☢

<10>

Java

compiled language

very large projects ✓

complex problems

small projects 🚫

general purpose

application server

frameworks ✓

resources 

scales very well ✓

Java API ✓

r there developers?

online job database

Java devs in Berlin

300+

$\langle 11 \rangle$

Perl

scripting language

quick solutions

system administration

mod\_perl

scales well

CPAN ✓

Perl devs in Berlin

65

<12>

PHP

scripting language

web development

very quick solutions ✓

easy ✓

spaghetti code 🌧️

scales well 🌧️

frameworks 🌂

PEAR ✓

PHP devs in Berlin

226

<13>

Ruby

scripting language

beautiful ✓

general purpose

framework ✓

Ruby on Rails

scales? 🌧️

Ruby devs in Berlin

54

<14>

ASP.NET

web development framework

Visual Studio

C

C++/C#

Visual Basic

quick solutions

complex solutions

scales well

Windows-only ☢

IIS-only ☢

ASP.NET devs in Berlin

32

<15>

JavaScript

not beautiful

not cross browser

framework ✓

jQuery

Prototype

script.aculo.us

Dojo

JavaScript devs in Berlin

157

<16>

Python

scripting language

general purpose

framework ✓

Django

Pylons

TurboGears

scales well

Google App Engine

Python devs in Berlin

39

<databases>

<17>

MySQL

M in LAMP

PHP

easy to use

web development ✓

small to very large

Flickr

Facebook

Wikipedia

MySQL DBAs in Berlin

49

< 18 >

Oracle

web development 🌂

very large databases ✓

business intelligence

small databases ☢

scales very well

\$\$\$

Oracle DBAs in Berlin

23

< 19 >

DB2

see Oracle

DB2 DBAs in Berlin

6

<20>

SQLite

db without server

embedded databases

Firefox

Skype

Mac OS X

web development 🌂

small projects

concurrent writes ☢

SQLite DBAs in Berlin

none

<gos & no gos>

<21>

portability 🌂

flexibility 🚫

performance 🚫

<22>

directory structure

img/

css/

js/

talking URLs

caching & load balancing ✓

<23>

source code control

SVN

Mercurial

Bazaar

Git

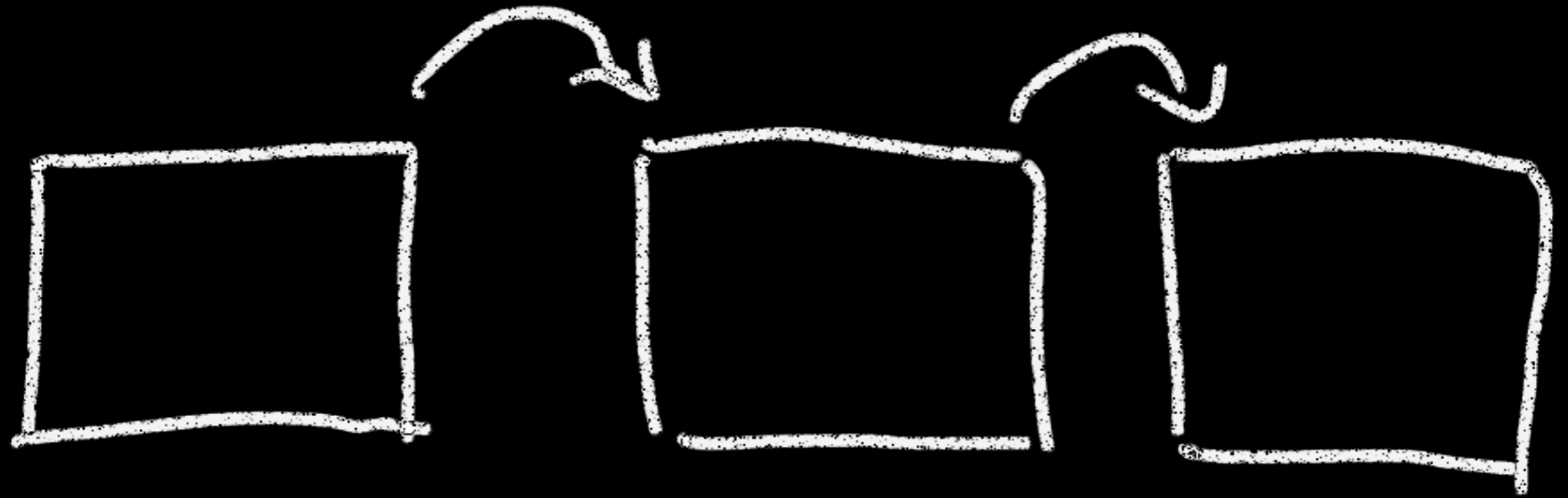
<24>

DSP model

development

staging

production

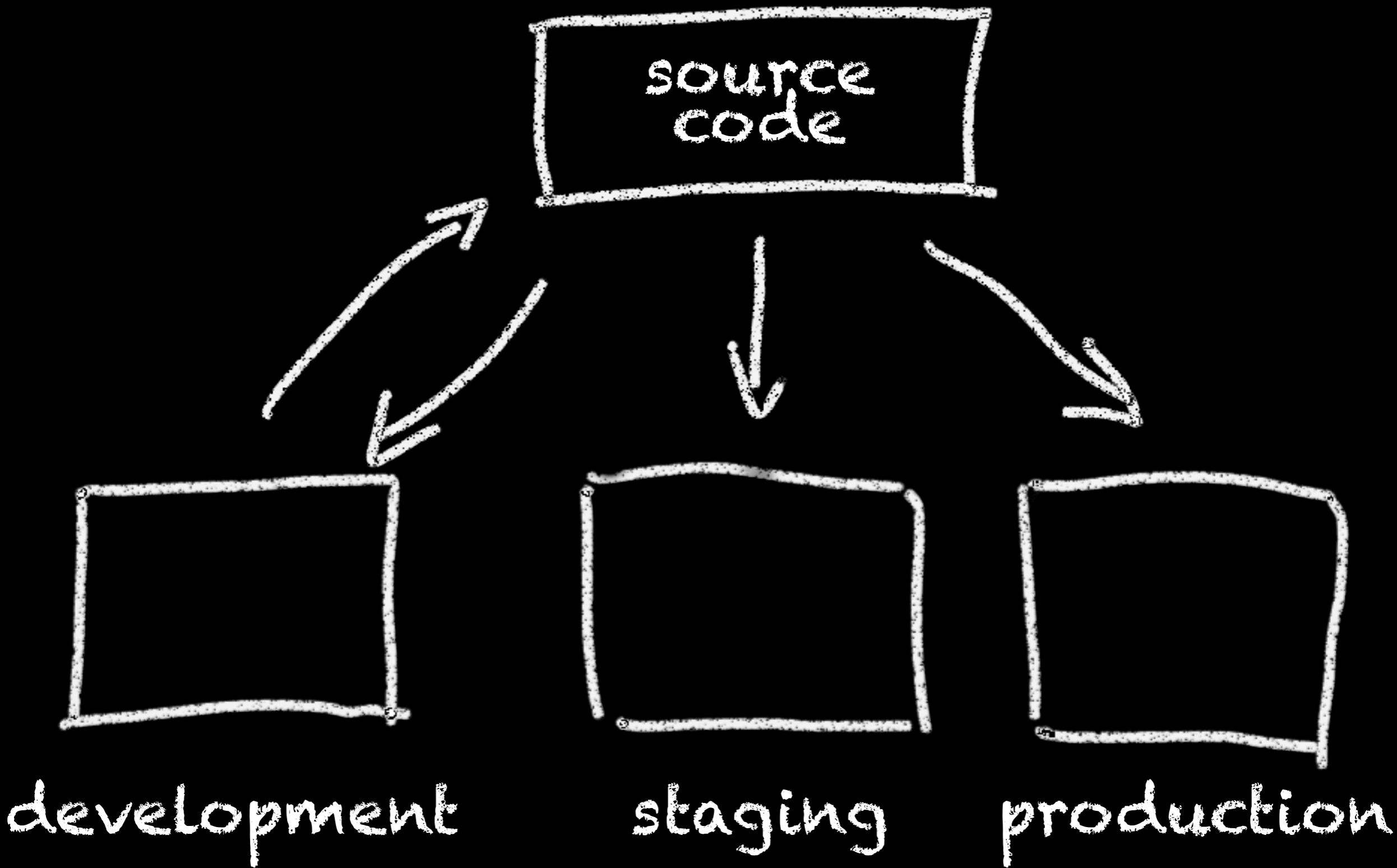


testing

<25>

automate deployment ✓

use repository to deploy



manual interaction 🚫

<26>

power of symlinks

checkout to production

old code/new code  
mixes up

checkout > new directory

htdocs -> new directory

<27>

automate build ✓

make

Ant

Rake

Hudson

manual interaction 🚫

<28>

monitor production

Munin

Nagios

MRTG

watch over resources

identify

memory problems

performance flaws

<29>

accelerate

scripting languages

eAccelerator

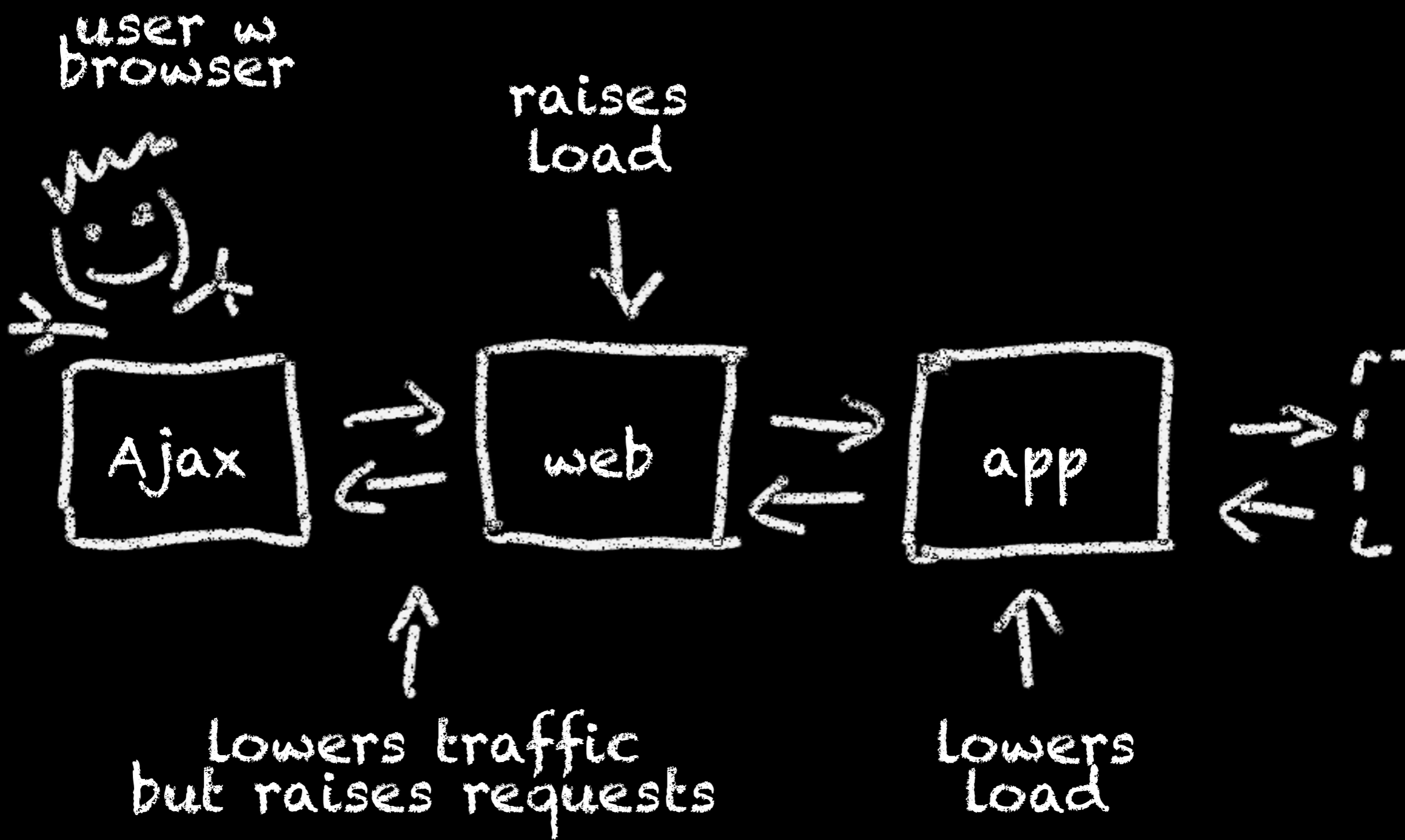
Psyco

Joyent

Minifi

<30>

reduce load with Ajax 🚫



security ☢

<31>

preloader with Ajax 

site more responsive ✓

too much preload

browser becomes clumsy ☢

<32>

store configuration  
in a database  
not in code

<33>

store session data in. . .

filesystem 🚫

no shared access

gets slow under load

database 🌂

shared access

stored permanently

raise load on db

memory ✓

memcached ✓

key/value store

very fast

shared access

<34>

be inaccurate  
where appropriate

online users?

```
SELECT COUNT(*) FROM session; ☠
```

cron -> memcached

<35>

use the power of HTTP

"Expires:" header

cache for a year

version number in filename

border\_top\_01.gif

border\_top\_02.gif

<34>

lovely favicon.ico

404 🚫

small & cacheable

"Expires:" header

<36>

cache, cache, cache

early stage of development

memcached

row-level database caching ☢️

complex objects

user profile

friend list

<some><rendered>... </html>

pre-populate

session-aware

unique keys

<37>

database normalization 🌂

joins are expensive

denormalization

time critical request

reduce joins

<38>

database abstraction layer ☢

flexibility 🌂

performance 🚫

<39>

stored procedures 🌧️

server limit ☢

<40>

put images in a db 

<scaling>

vertical

horizontal

<41>

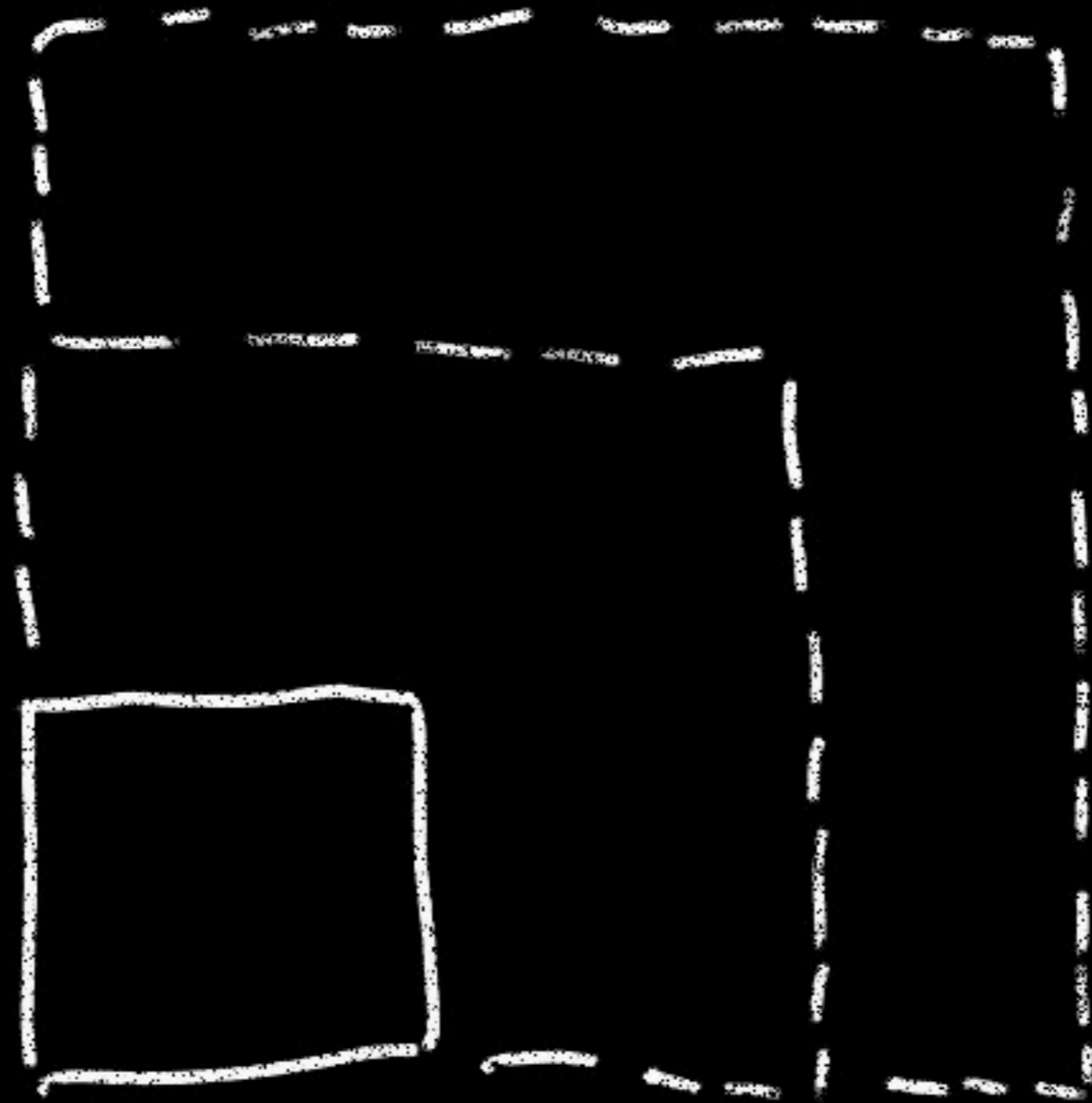
let's vertical scale

add RAM

add CPUs

more hard disks

vertical scaling



easy

physical limits

<42>

let's horizontal scale

add more machines

# horizontal scaling

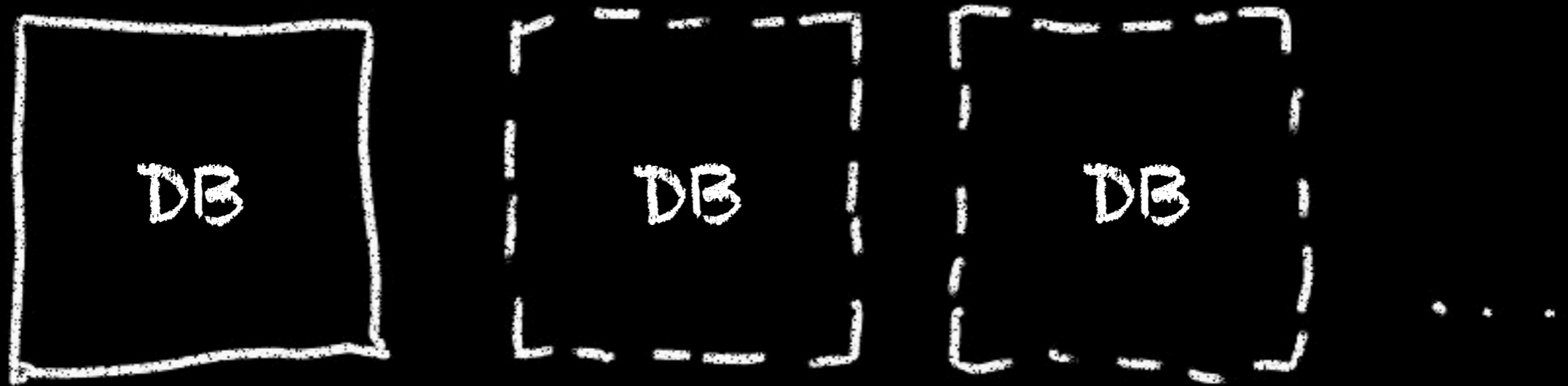


...

# horizontal scaling



# horizontal scaling





W

W

W

W

A

A

D

problem

how to connect?



?

W

W

W

W

?

A

A

?

D

by application logic

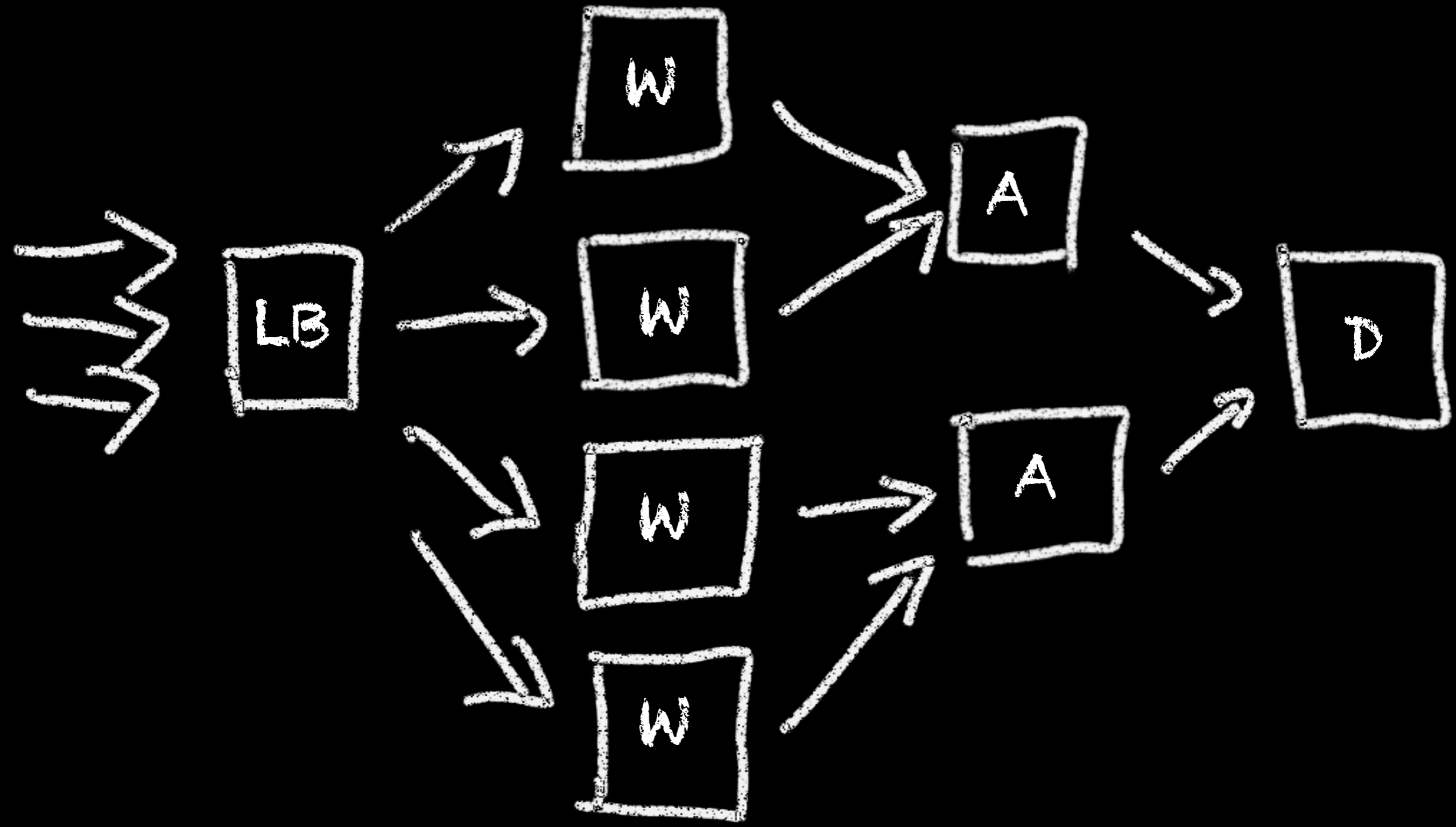
by configuration



?



by load balancer



distribute requests

by count

by traffic

by load

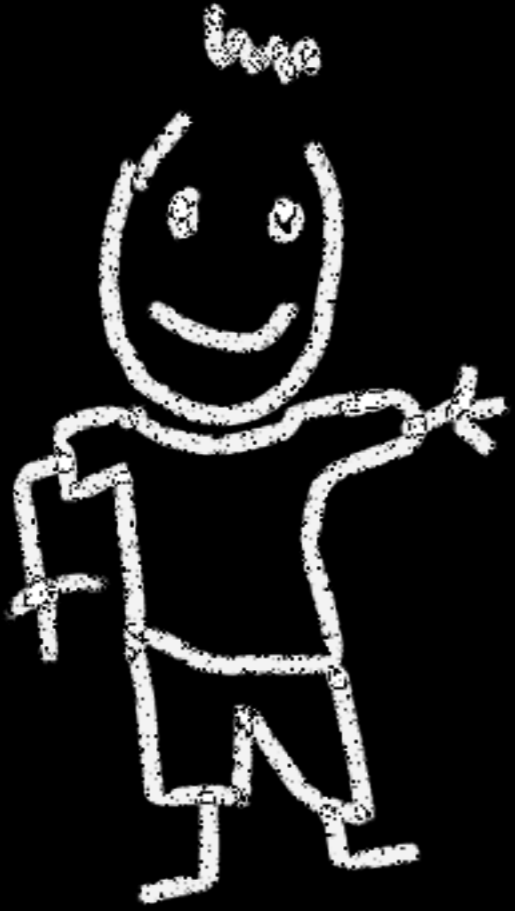
fault tolerance

2 types

layer 4

transport layer

TCP/UDP



LB



?

?

?

?

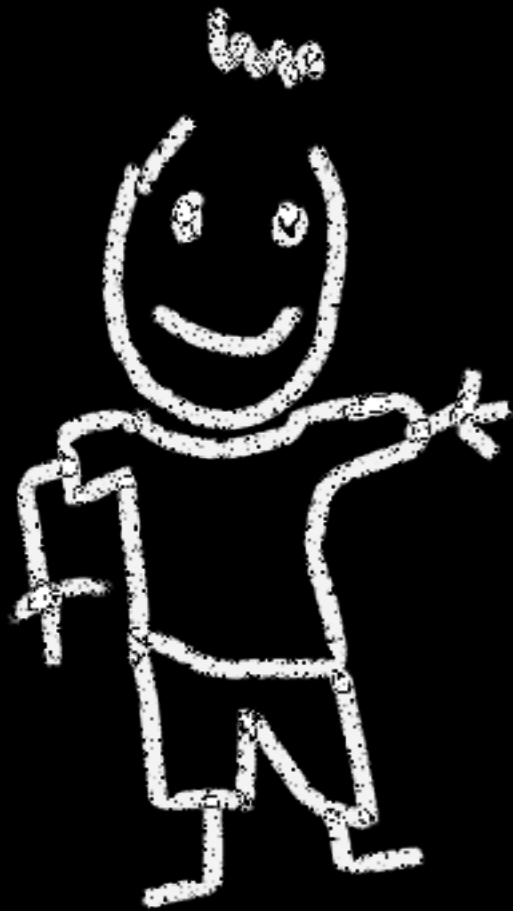
IPVS ✓

layer 7

application layer

knows about HTTP

session aware



USER W  
SESSION

8

LB

W

1, 2, 3

W

4, 5, 6

W

7, 8, 9

W

10, 11, 12

Apache ✓

<thanks>

codebits

lisboa

YOU!

Questions?

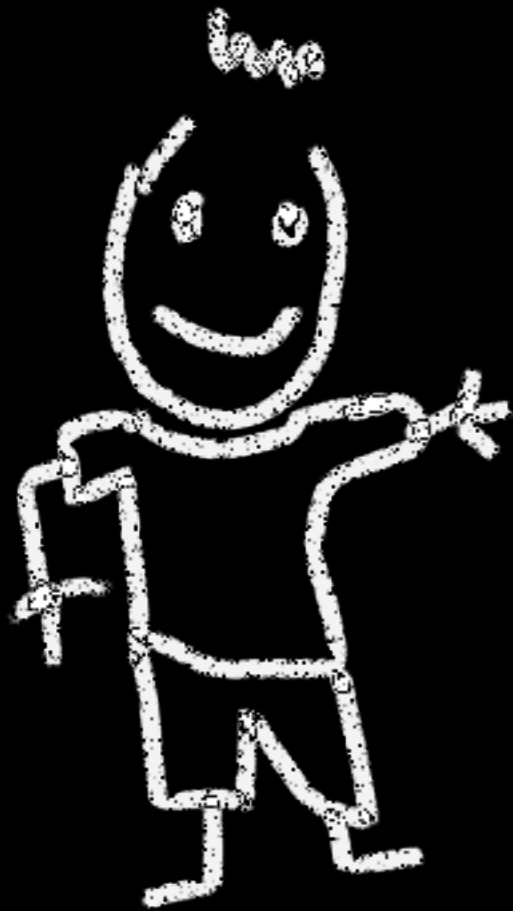
[oswald@sun.com](mailto:oswald@sun.com)

<backup slides>

<43>

layer 7 is complex

session awareness



USER W  
SESSION

8

LB

W

1, 2, 3

W

4, 5, 6

W

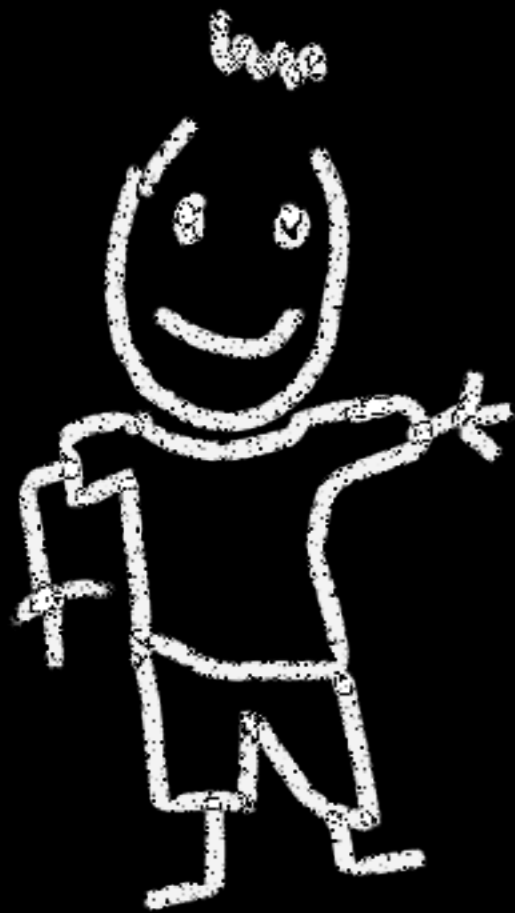
7, 8, 9

W

10, 11, 12

layer 4

central session store



USER W  
SESSION

LB

W

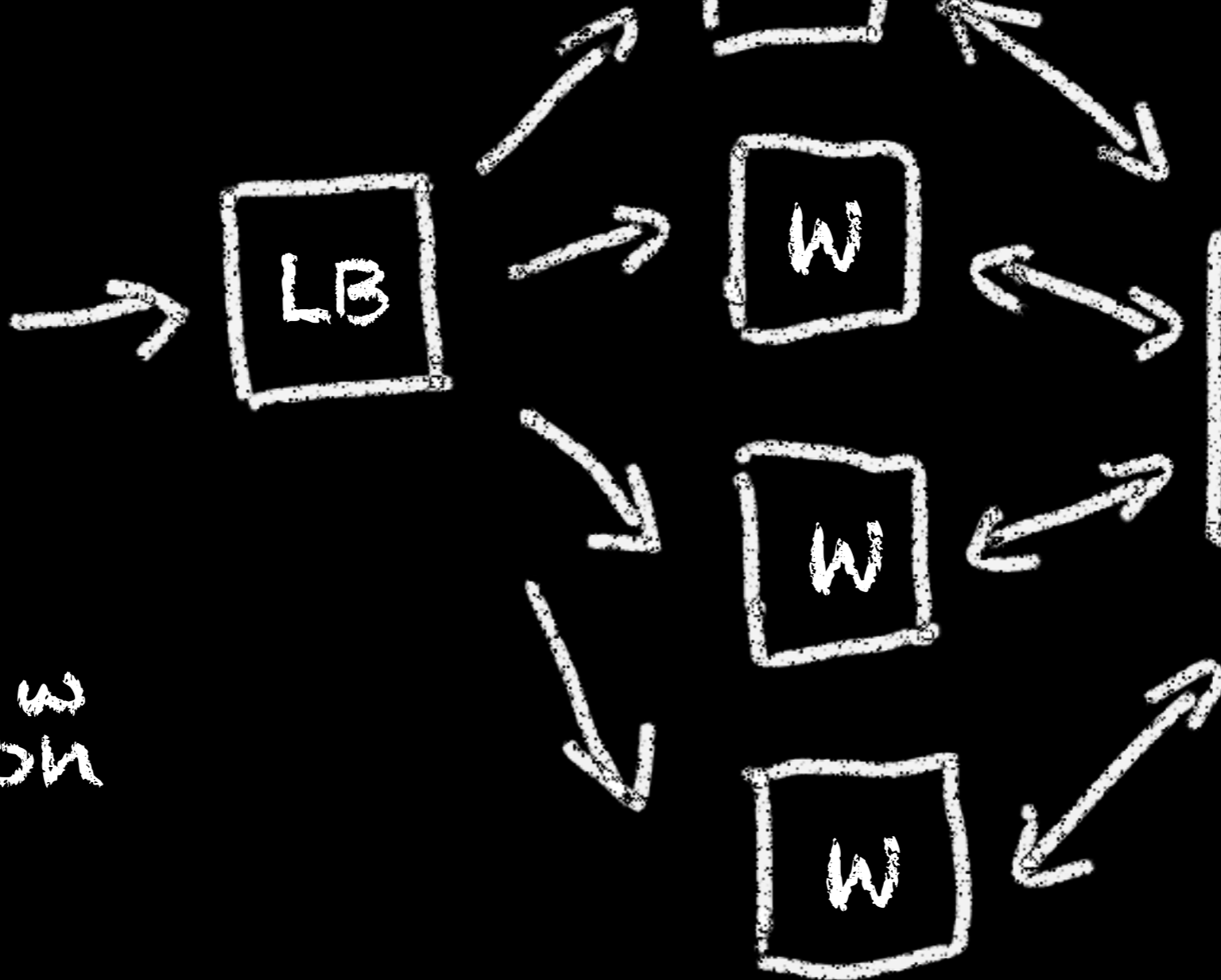
W

W

W

central  
session  
store

1,2,3,4,  
5,6,7,9,  
10,11,12



memcached

PHP

```
session.save_handler = memcache  
session.save_path = "tcp://serverM:11211"
```

Thanks!