Beat the Devil

towards a Drupal performance benchmark

http://drupal.org/user/770300
@rodricels

http://drupal.org/user/8859
@perusio

http://drupal.org/user/39078
@NITEMAN_es

We'll kill a lot of sacred cows

the murder weapon will be

Occam's razor

which amounts to the
principle of least effort

laziness is good

hopefully in the end you'll be puzzled

good things happen to
puzzled people who
obsess

simplest
laziest
easiest


performance is for
everyone

do you have
performance issues?

http://www.flickr.com/photos/irisheyes/148134965/sizes/o/in/photostream/

mathematics is
for everyone

request per second

vs

seconds per request

MTTR

Mean Time To

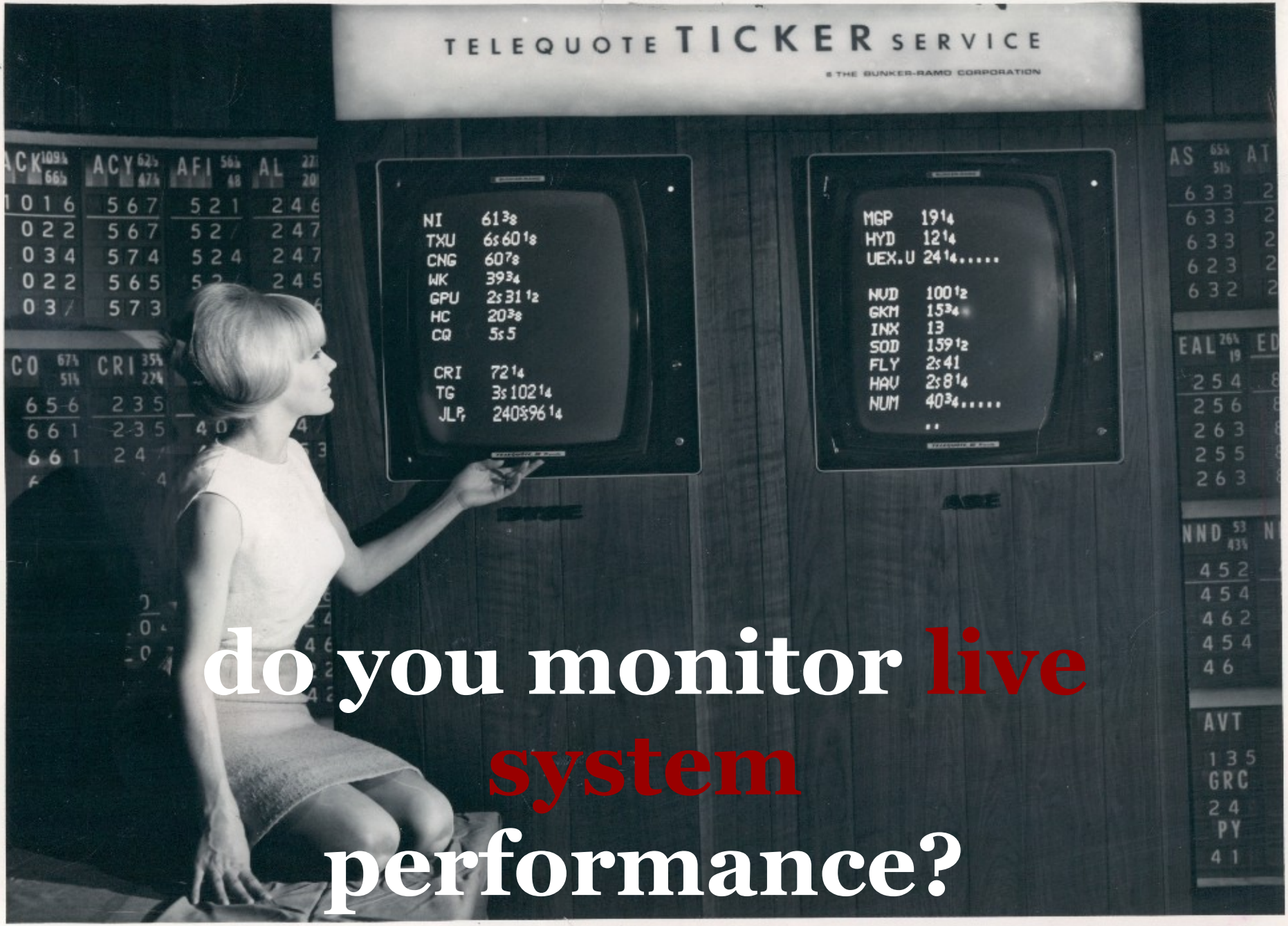
Recovery

vs

MTBF

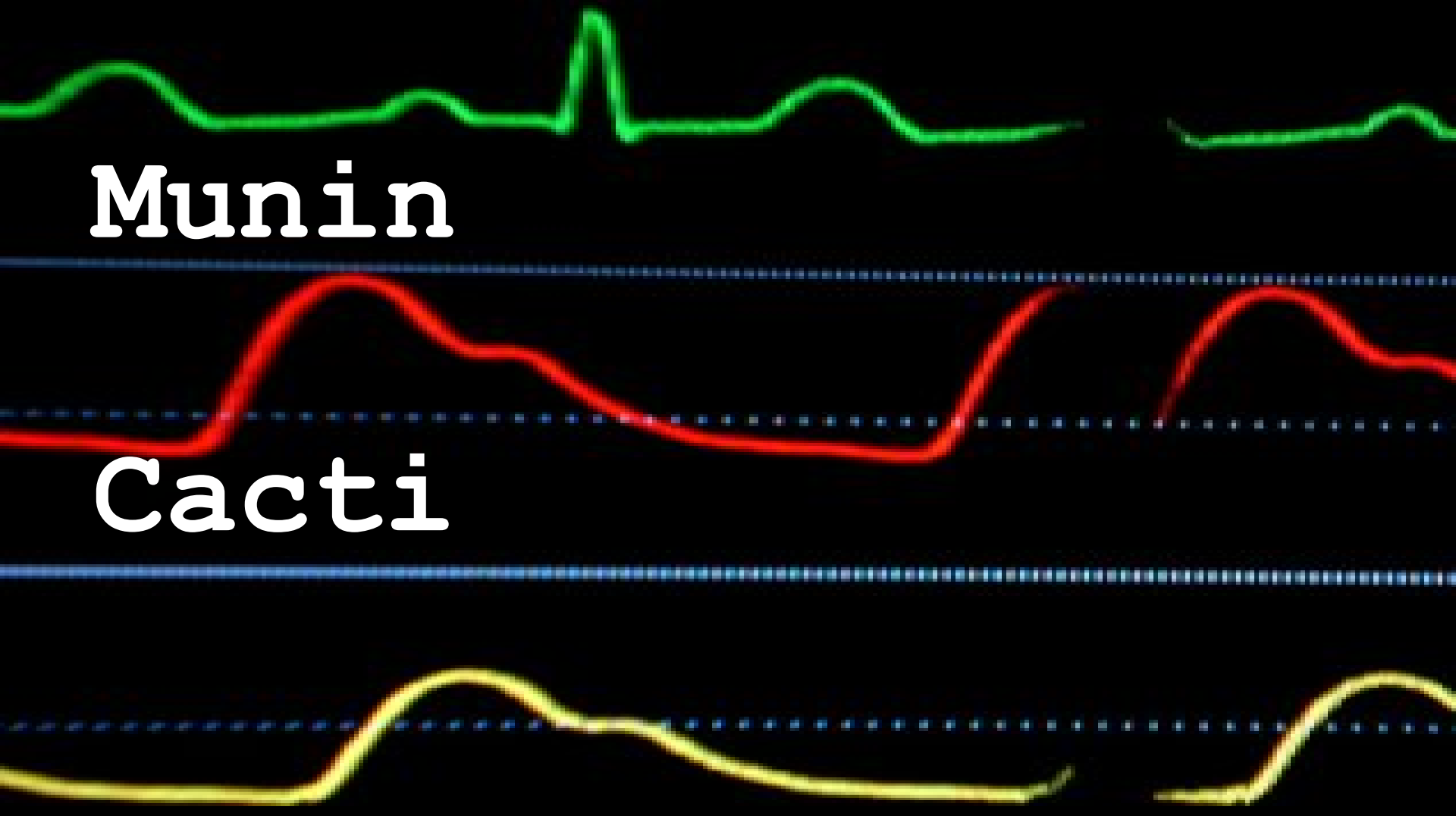
Mean Time Between

Failure



do you monitor live system performance?

live monitoring tools



drupal accesslog table
has tons of useful data

beware the logs

`logstash` / `graylog`

aggregate them


economics
costs per
request/volume/peek
operational costs
Return On Investment

we're talking about $peed

$peed = cost_effective_performance

slowness
downtime
operations
costs you money

it's not the
tools you have

it's the use you
make of them

have you benchmarked your Drupal?

complex is easy and fragile

simple is hard and resilient

Keep It Simple Stupid

Know your stack
Know your targets
Monitor your performance

find where the problems are
don't fix things that aren't broken
the worse the first
one change a time
&
keep a log of your actions

dissecting

frontend vs **backend**

backend
static vs **dynamic**

dynamic
processing vs **data gather**

beyond development & operations

good developers are those that take **all aspects** in consideration

test systems

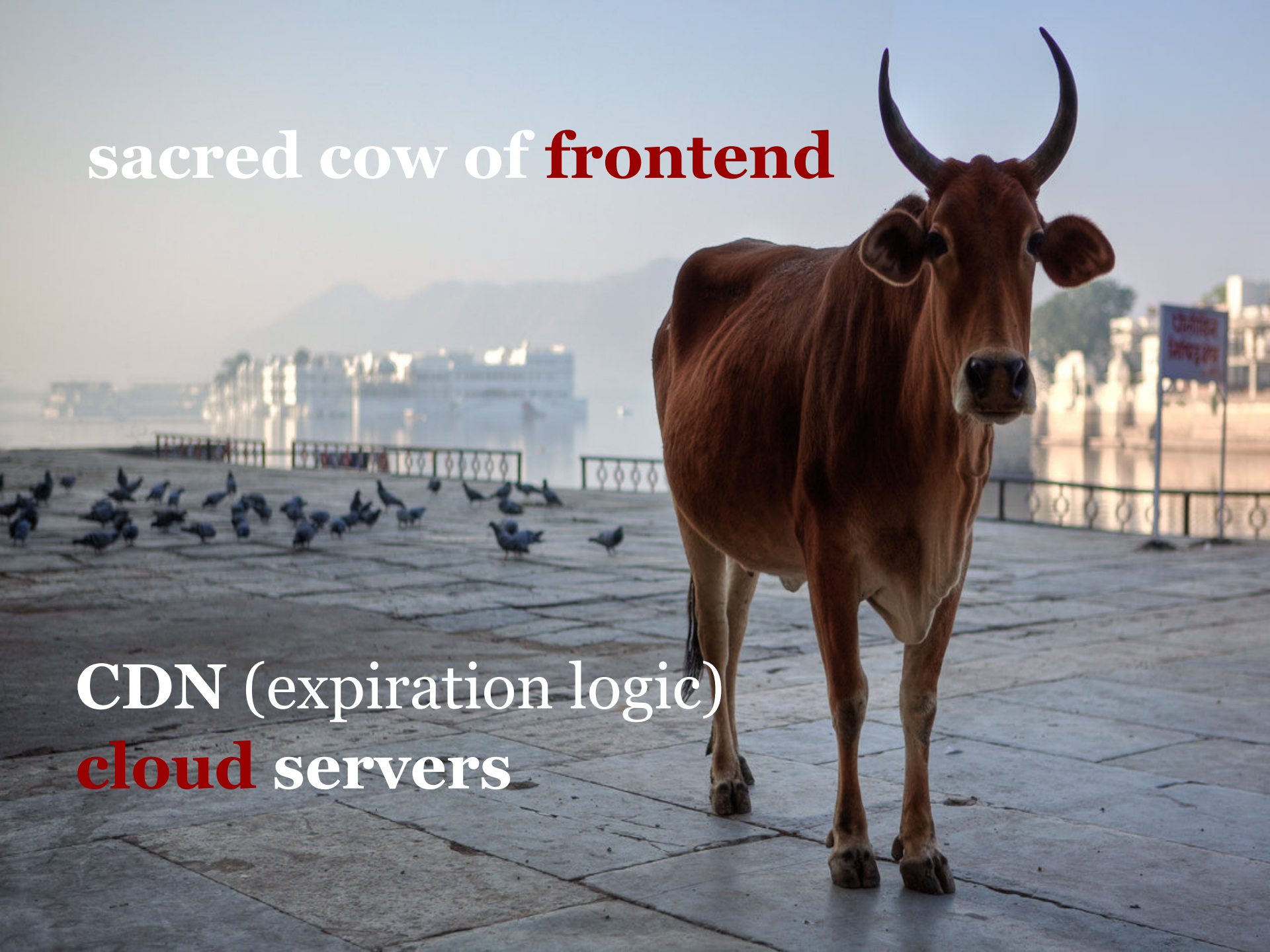
keep everything as close as possible to production

test systems

take advantage of
configuration management
to reproduce your live
infrastructure

test systems

some software has its
performance directly
tied to the number of cores
and/or
the amount of RAM

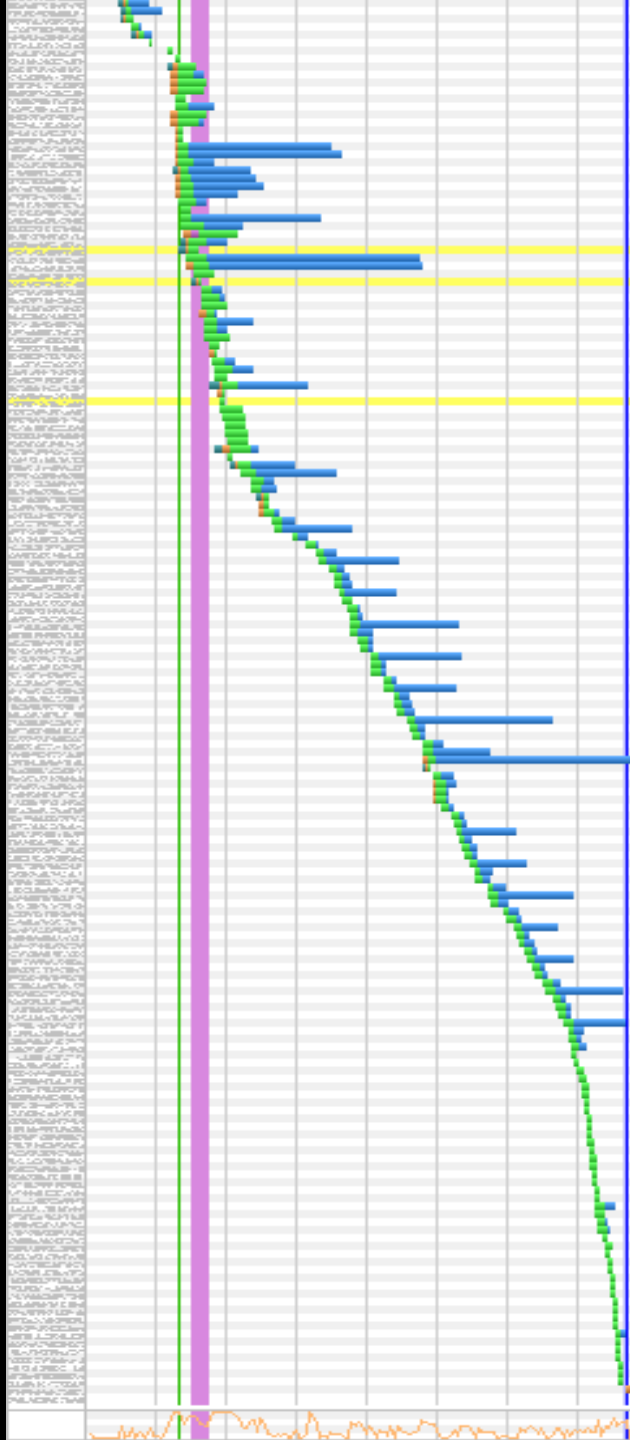

sacred cow of frontend
CDN (expiration logic)
cloud servers

frontend vs backend

number of req. / parallelization

blocking events

data side: DNS resolution
data side: download time
data side: size / weight
order matters



frontend vs backend tools

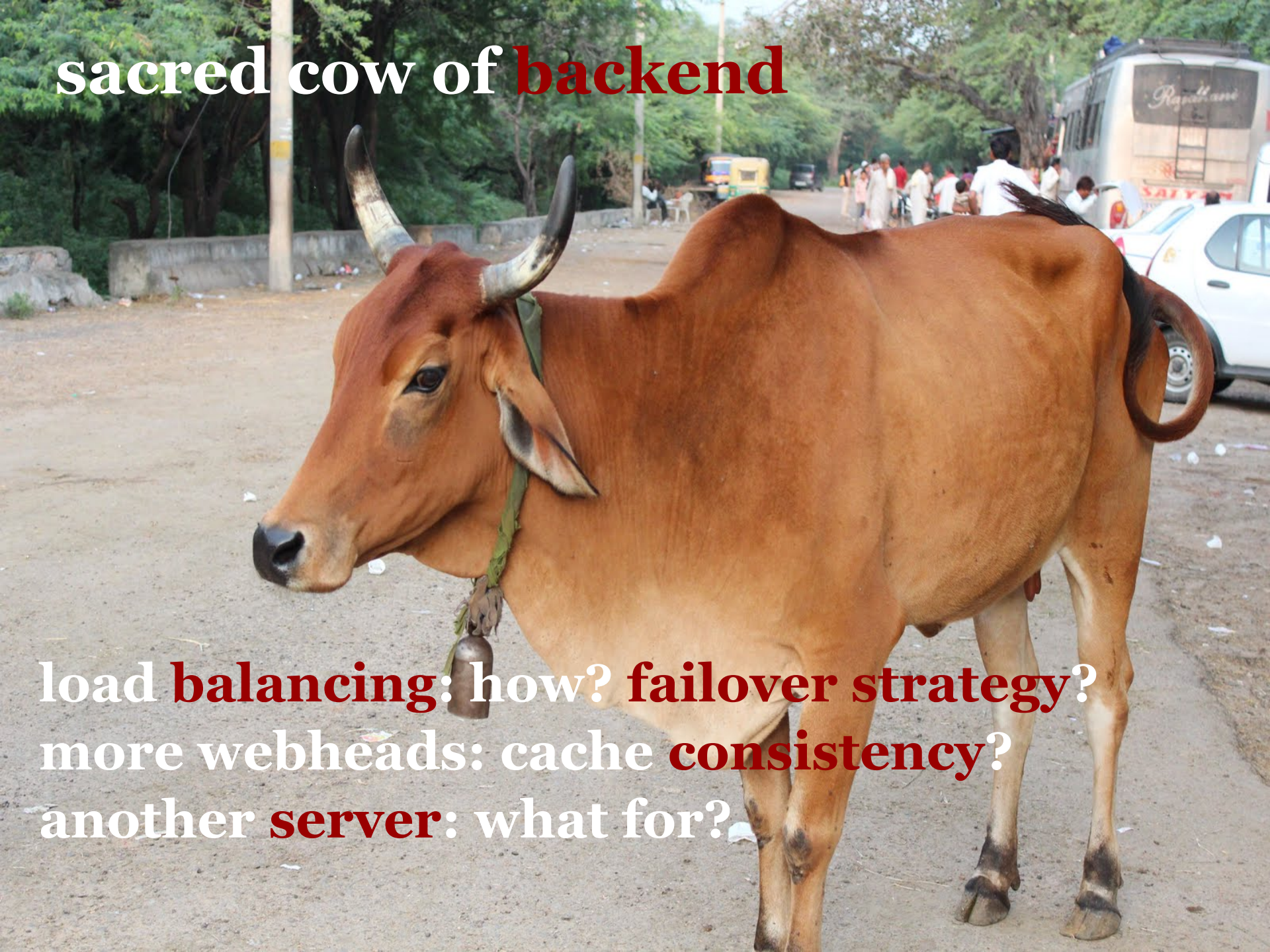
Firebug

Chrome tools

Google Speed Tracer

Yslow

Web services [*] [*] [*] [*]


sacred cow of backend
load balancing: how? failover strategy?
more webheads: cache consistency?
another server: what for?

backend: static vs dynamic

anonymous vs registered

cacheable vs non cacheable

HTTP benchmarking tools

~~ab~~

Jmeter

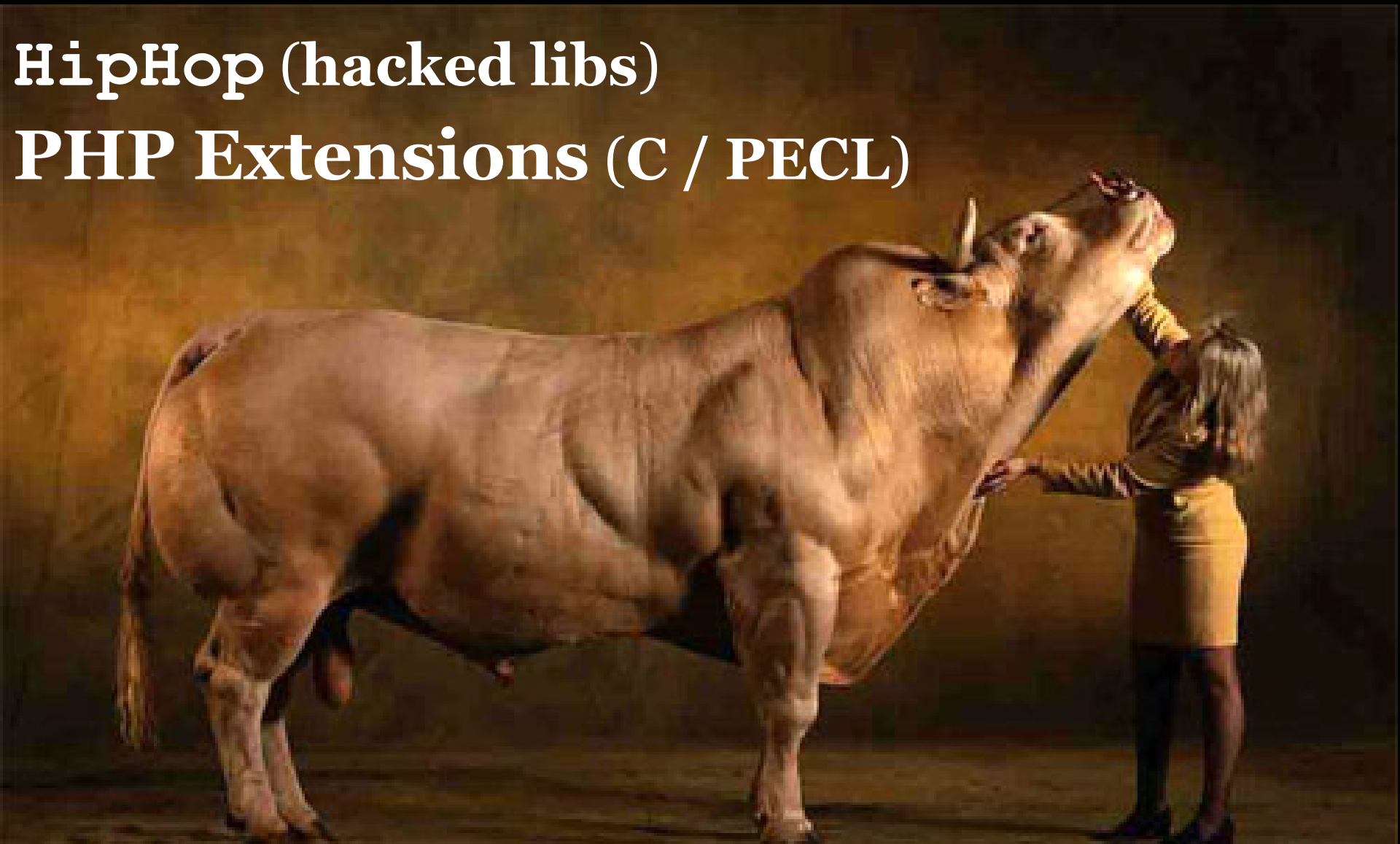
Httperf

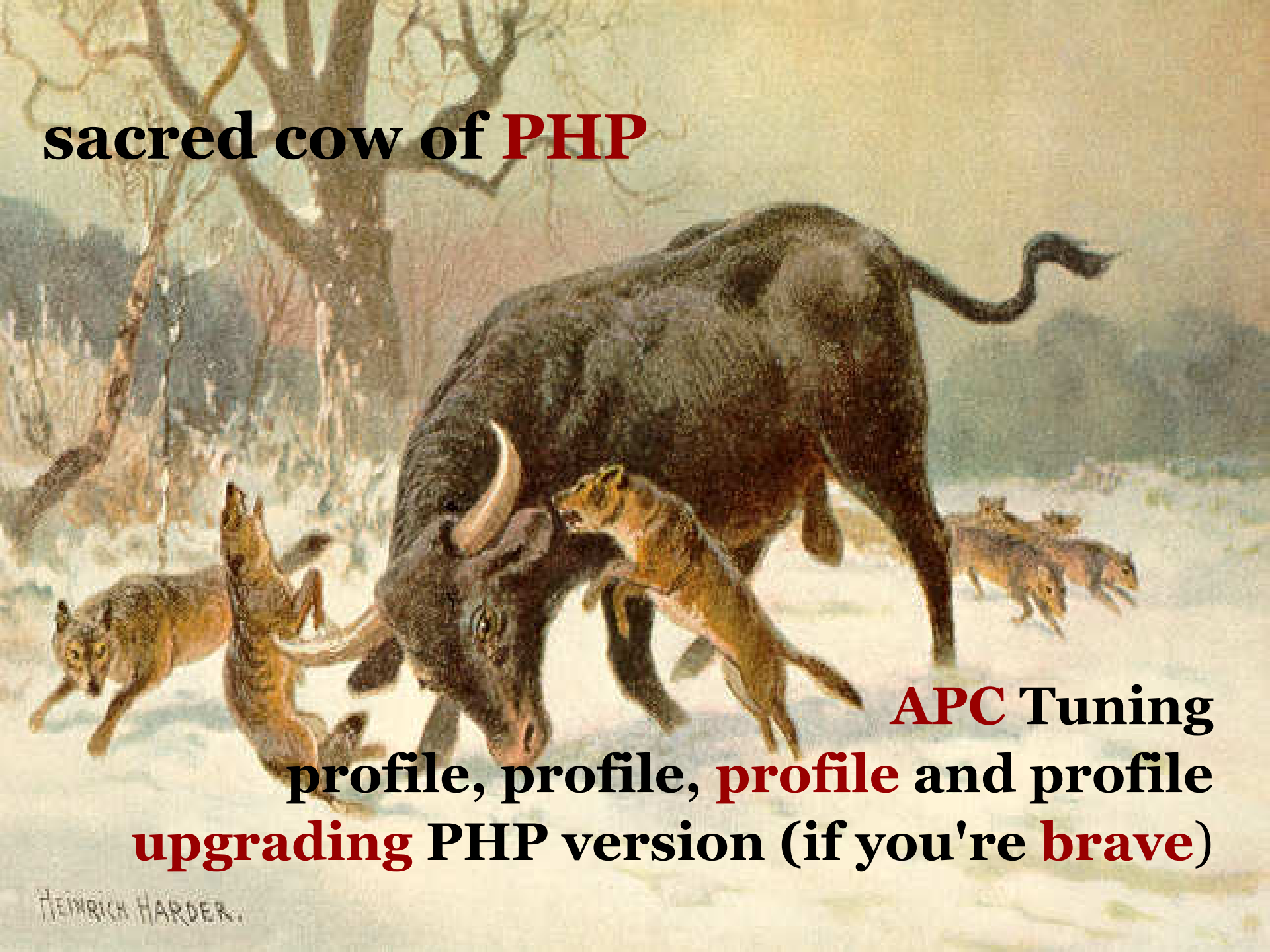
wrk

Gatling

TCP copy

sacred cow of PHP




sacred cow of PHP
APC Tuning
profile, profile, profile and profile
upgrading PHP version (if you're brave)
HEINRICH HARDER.

dynamic
processing vs data gather

locks

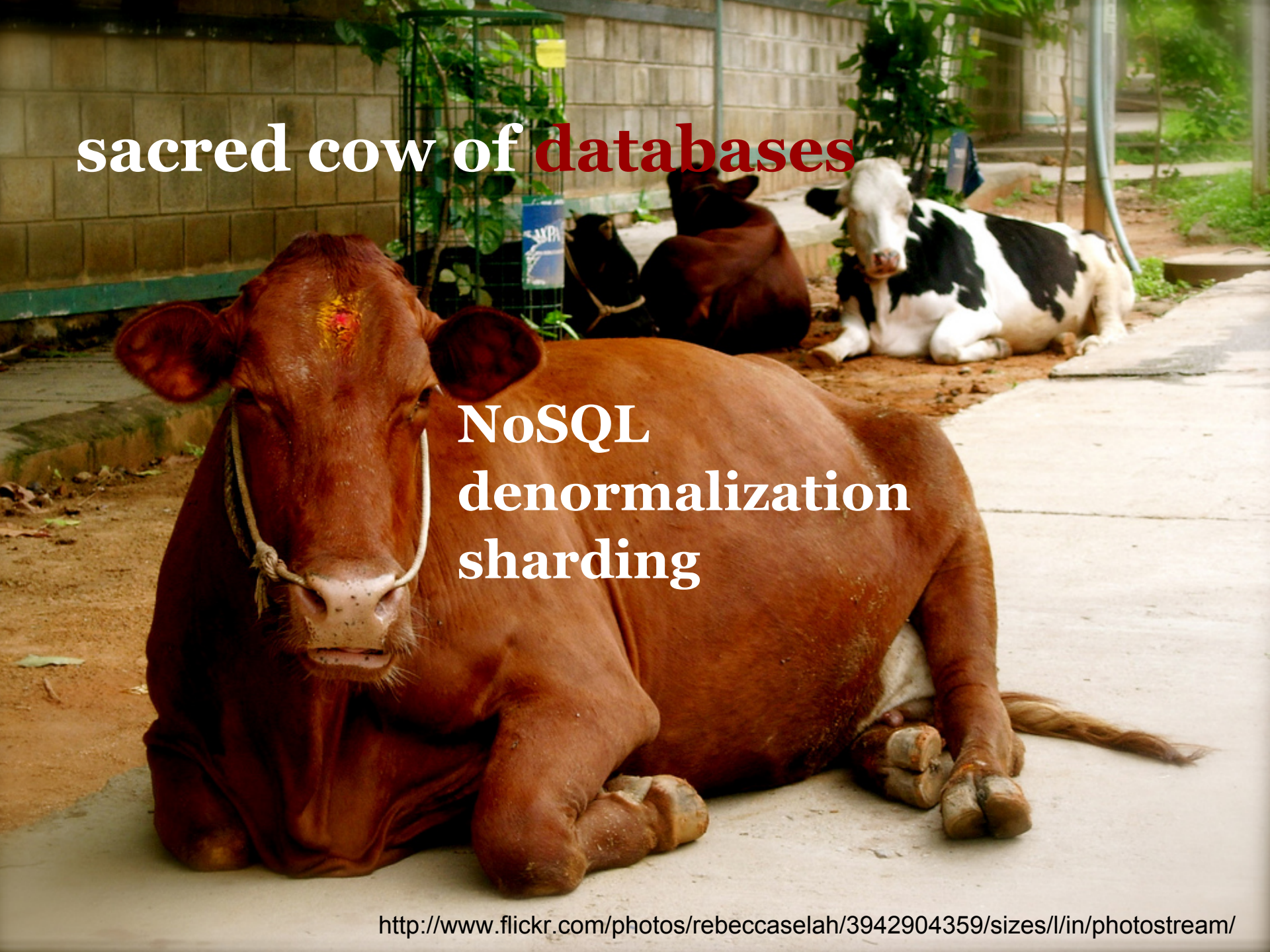
difficult to dissect

devel

benchmarking / profiling PHP

Xdebug
Webgrind
Xhprof

don't even try without an opcode cache
& remember the hard disk


sacred cow of databases
NoSQL
denormalization
sharding
http://www.flickr.com/photos/rebeccaselah/3942904359/sizes/l/in/photostream/


sacred cow of databases
NoSQL => helps with writes
denormalization => helps with HUGE DBs
sharding => helps with Big DBs

MySQL Tune & Benchmarking tools

https://tools.percona.com/

Percona toolkit [*] (formerly maat-kit)

tcpdump + Percona toolkit

Logs are vital!

Every engine has strongs & weakness

Better on bare metal!

root hardware causes

network & DNS play a role

static content depends mainly on I/O

dynamic content (php) depends on CPU

database server mainly depends on RAM

extra balls:

Drupal known issues (DB): Watchdog, sessions, accesslog… history…

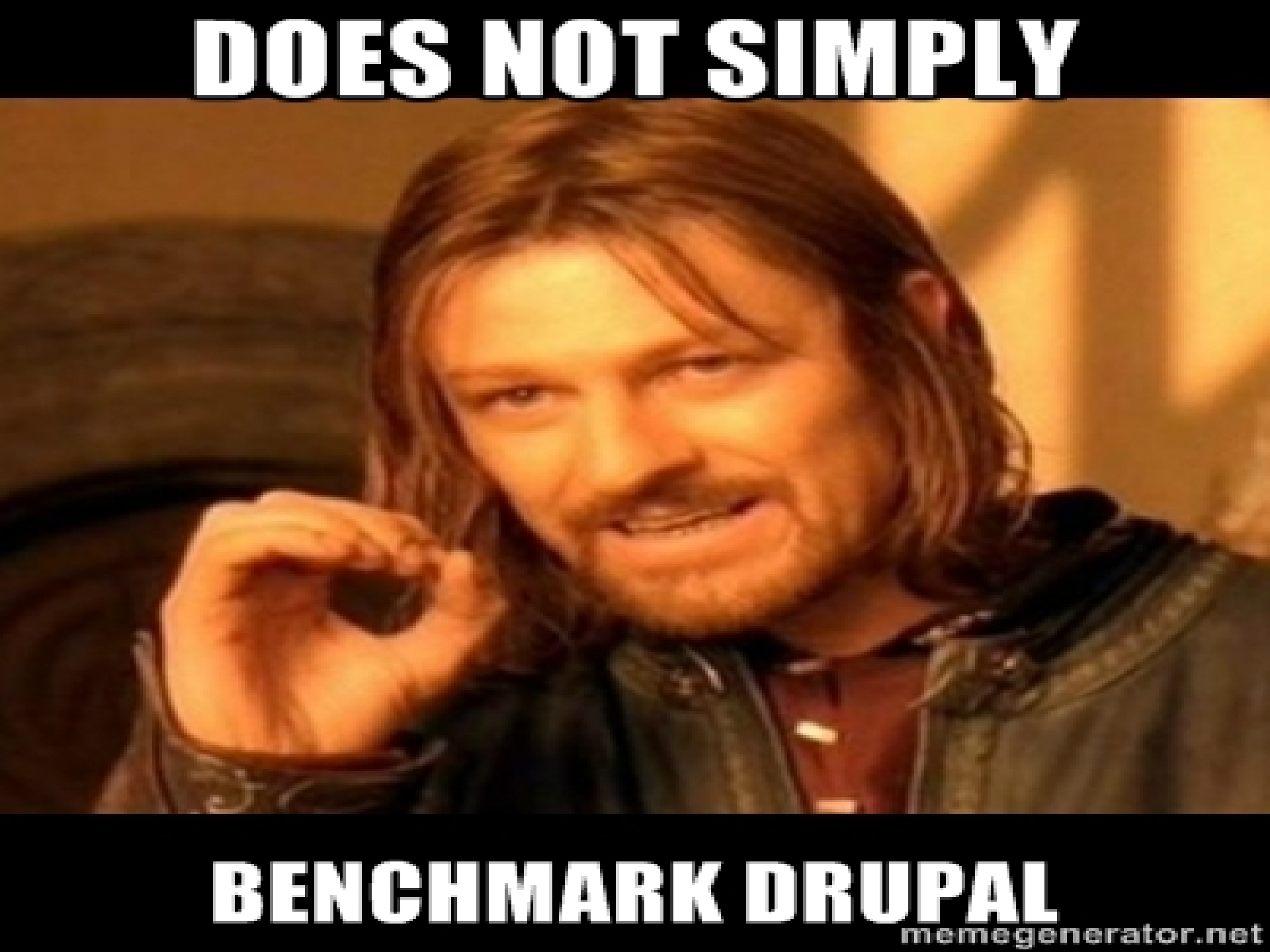
Please do stress, load & stability tests regularly in your live system

This is our way, what is yours?

(no cows were harmed in the making of this presentation)

¿Questions?

Pablo Picasso [speaking of computers]:
"But they are useless. They can only give you answers."


DOES NOT SIMPLY
BENCHMARK DRUPAL
memegenerator.net