Erlang as a cloud citizen

Paolo Negri @hungryblank

wooga

world of gaming

1 day at  wooga
world of gaming

- 10 millions players
- 2 billions game server requests (http)
- 20 devops people

Cloud

"A cloud is made of billows upon billows upon billows that look like clouds.

As you come closer to a cloud you don't get something smooth, but irregularities at a smaller scale."

Benoît B. Mandelbrot

http://www.flickr.com/photos/nirak/644336486

AWS Cloud


· Close
EC2 is pretty cool if you don't need network IO, disk IO or cpu
40
RETWEETS
16
FAVORITES
10:04 PM - 9 Feb 12 via Twitter for Mac · Details
EC2 is pretty cool if your business requires a lot of money burning
Hide conversation

This talk will answer

- Why building a system targeting the cloud?
- How many EC2 instances do you need to respond to 0.25 billion uncacheable game reqs/day?

15 months ago...

1st cloud hosted project, lessons learned

1

Pushing live the 60th application server?

not different from adding the 6th!

push a button

1st cloud hosted project, lessons learned

2

local network/local disk are low performance general purpose tools

(nothing to do with ad hoc data center solutions)

1st cloud hosted project, lessons learned

3

Complete automation is cool

Ease of adding hosts/automation can lead to bloated infrastructure

1st cloud hosted project, points of pain

- A lot of inefficient app servers (as per tweets)
- Much effort to scale up/maintain databases (mySQL & Redis)
- Expensive, not crazy expensive, but expensive

Why trying again?

http://www.flickr.com/photos/kky/704056791/

Uncertainity

- will we reach100K or 3 millions users?
- 3 million users in 2 weeks or 12 months?
- cheat tool released 1h ago => single game call up 5000%
- weekly releases, new feature performance impact?

the cloud

- standard units (instances) of computing capacity
- a network connecting all instances
- an API to provision/dismiss instances

the cloud

Sounds like a good framework to compose computing capacity

Why didn't work as a framework to compose throughput?

Scaling in the cloud
the recipe

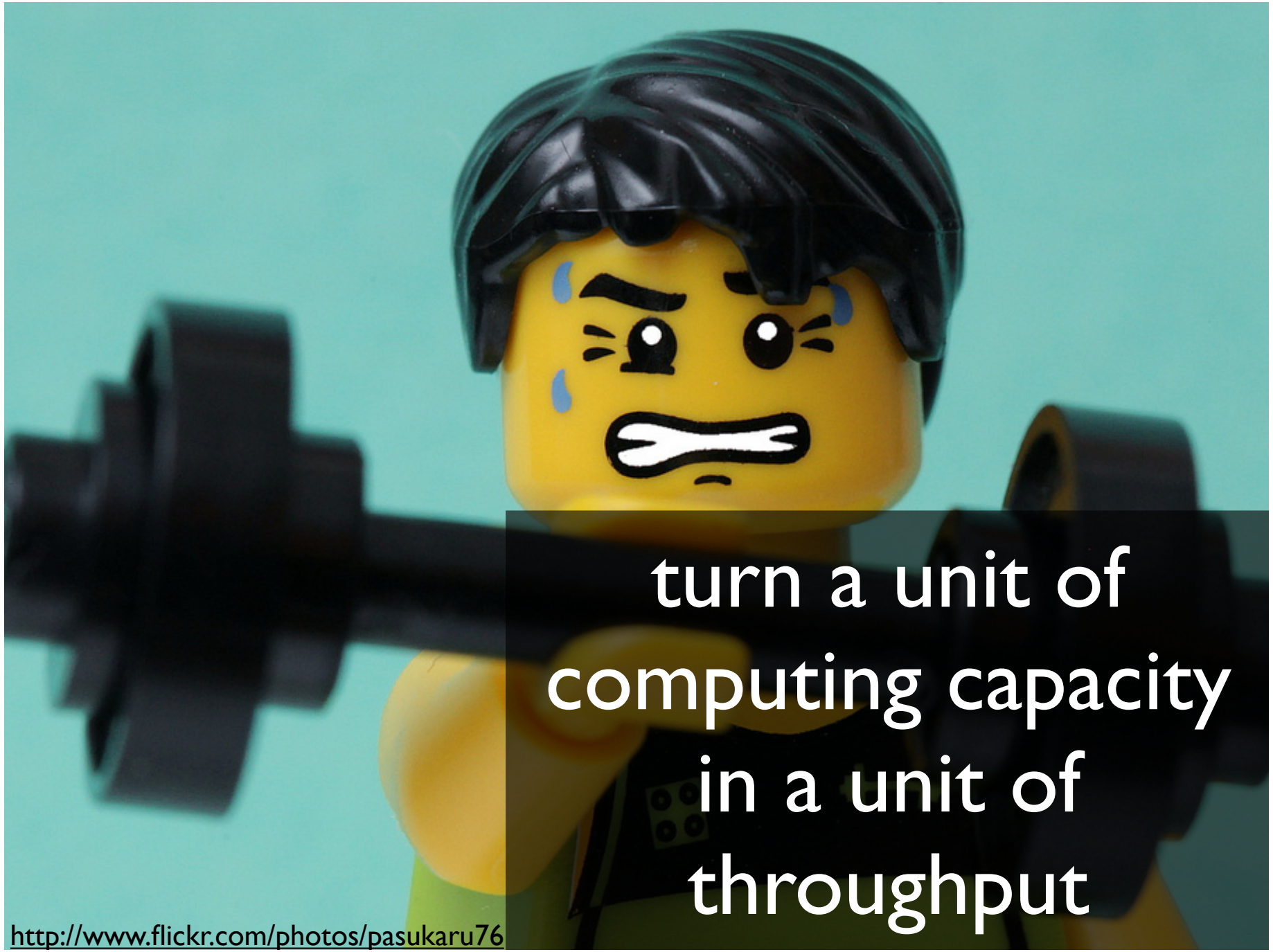
CLOUD: composable units of computing capacity

+

DEVELOPER: turn a unit of computing capacity in a unit of throughput

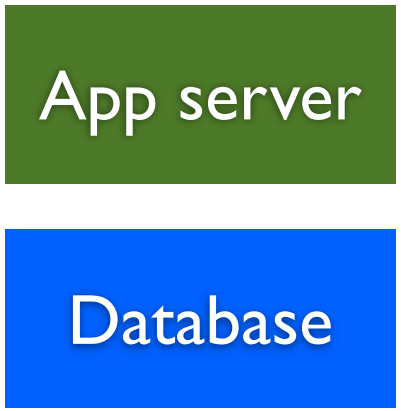
=

composable throughput, a plan for scaling BIG

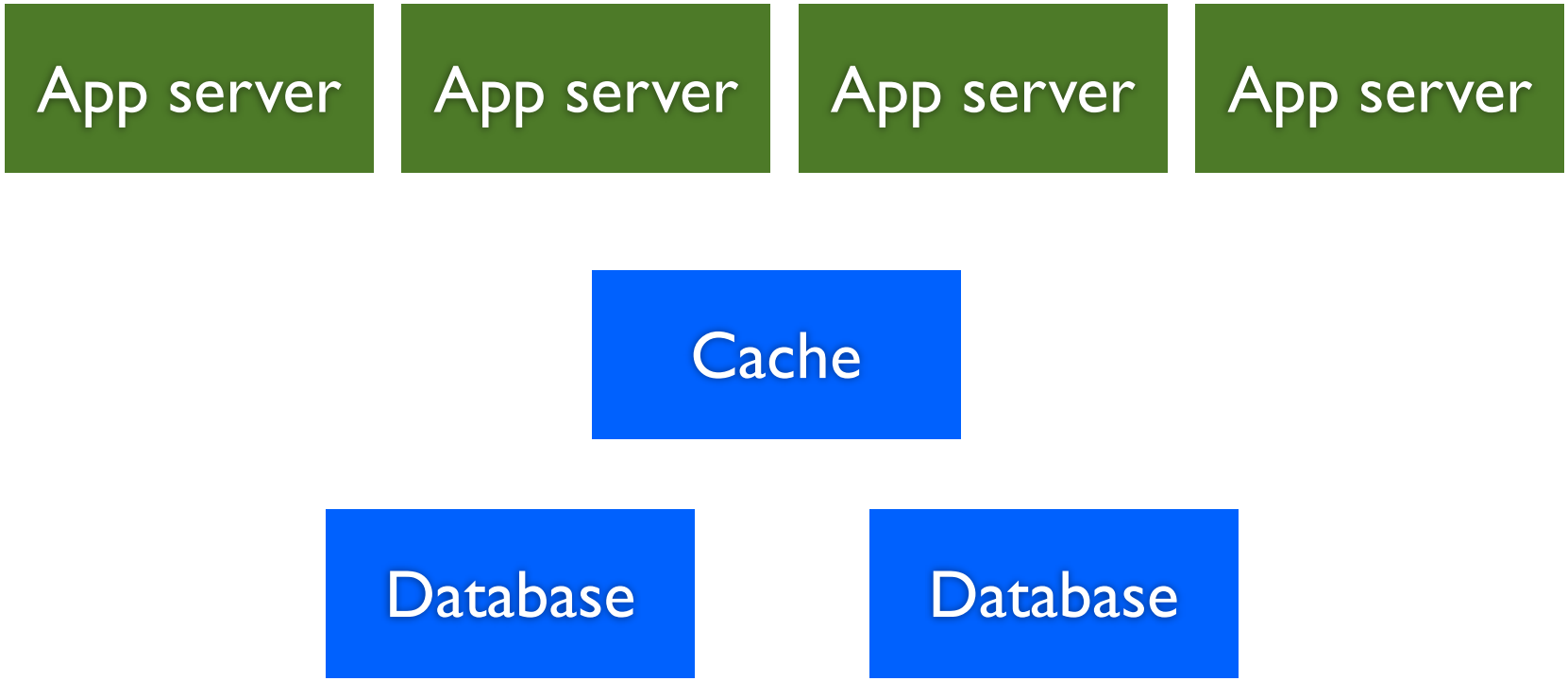


http://www.flickr.com/photos/pasukaru76

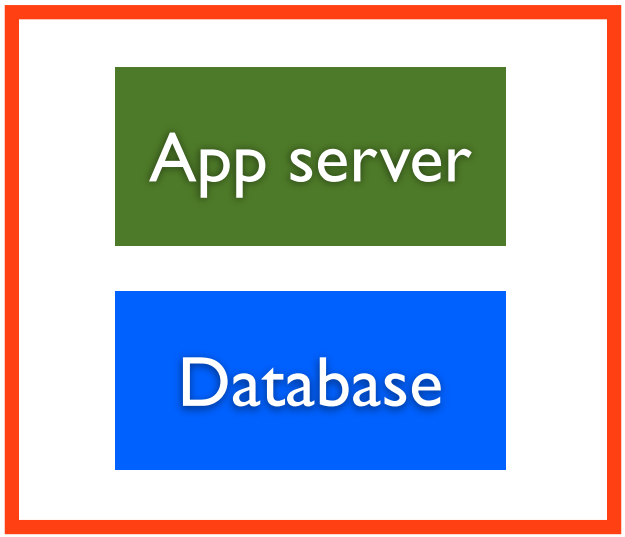
Unit of throughput, where?



Unit of throughput, joke?

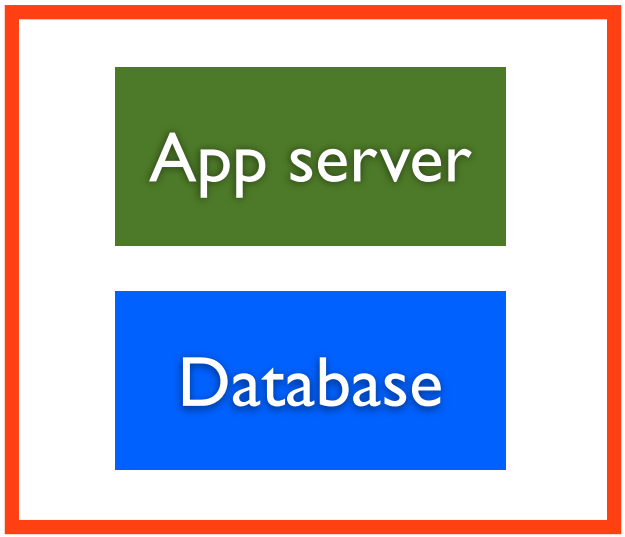


Unit of throughput?



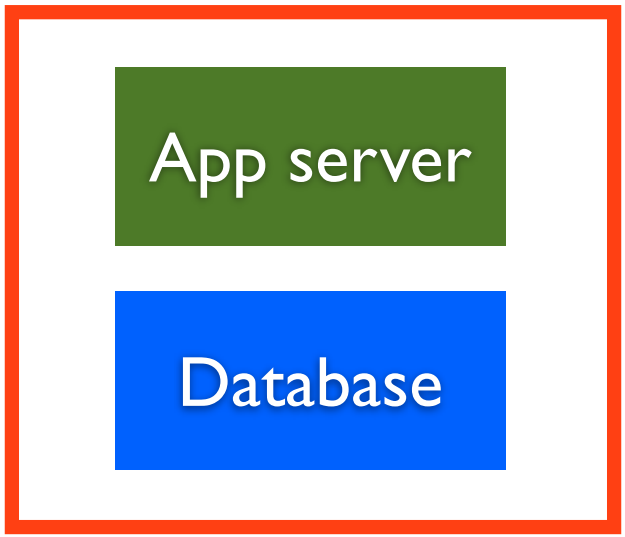
No unit!

Unit of throughput?



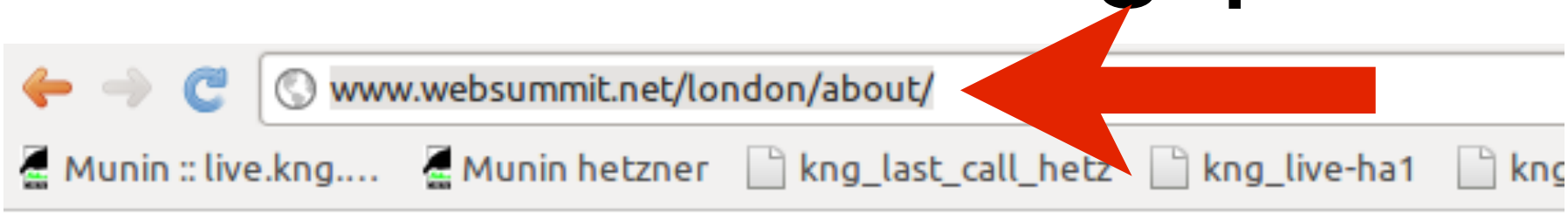
Tightly coupled throughput?

Unit of throughput?



Monolithic throughput?

Monolithic throughput



Error establishing a database connection

Monolithic throughput

- likes monolithic infrastructure!
- scales well vertically
- wants screaming fast stack (network, disks...)
- any performance glitch impacts the whole system

Tightly coupled throughput

+

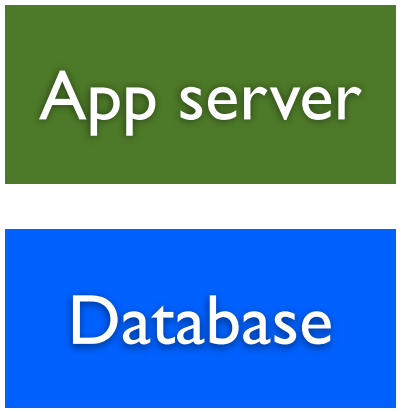
loosely coupled hardware

(like cloud)

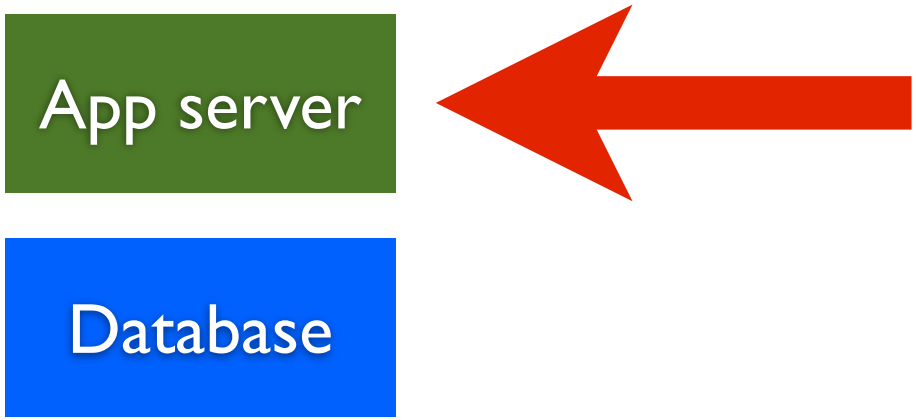
=

frustration

Who leads the tightly coupled dance?



Who leads the tightly coupled dance?



The stateless application server!

Stateless application servers guarantee one thing...

which?

Data is never
where you need it

And another one...

If you can feed them
data fast enough...

they'll choke on
garbage collection

We measure memcache
HIT / MISS

why app servers need
to be 100% MISS?

Where's the best knowledge about hot/ cold data?

Even the reverse makes more sense

1. pick your data up
2. go in the stateless app server

What Went Wrong?

He can tell you!

- Rich Hickey
- Clojure author

"...If not in Erlang which I think has a complete story for how they do state"[1]

[1] Value Identity State @0.27

http://goo.gl/Zdjv0

http://www.flickr.com/photos/ghoseb/5120173586

Most languages and runtimes don't have a safe solution for concurrent, long lived state

Erlang stands out as an exception in this panorama

Erlang...

Processes are the primary means to structure an Erlang application.

wikipedia

Erlang + OTP

Generic Server Behaviour

A generic server process
(gen_server) implemented
using this module...

otp documentation

Erlang + OTP
Generic Server Behaviour

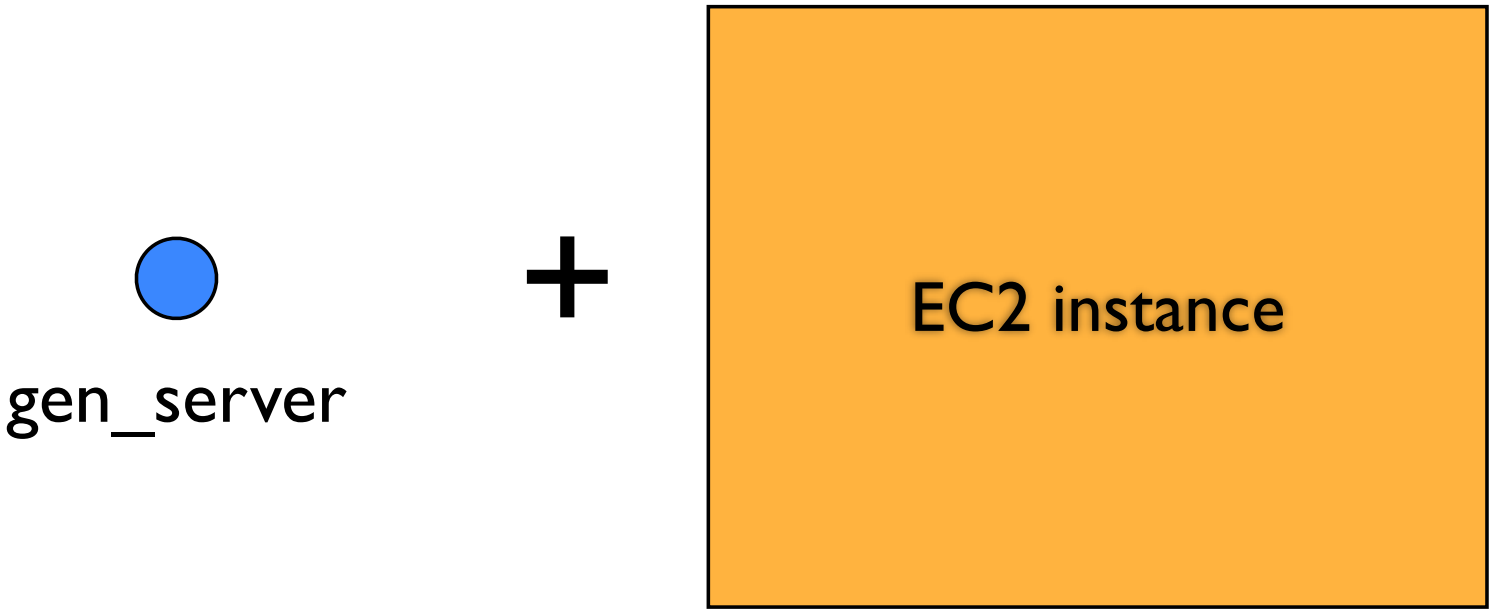
```
handle_call(_Request, _From, State) ->
    {reply, ignored, State}.

handle_cast(_Msg, State) ->
    {noreply, State}.

handle_info(_Info, State) ->
    {noreply, State}.

terminate(_Reason, _State) ->
    ok.

code_change(_OldVsn, State, _Extra) ->
    {ok, State}.
```

gen_server

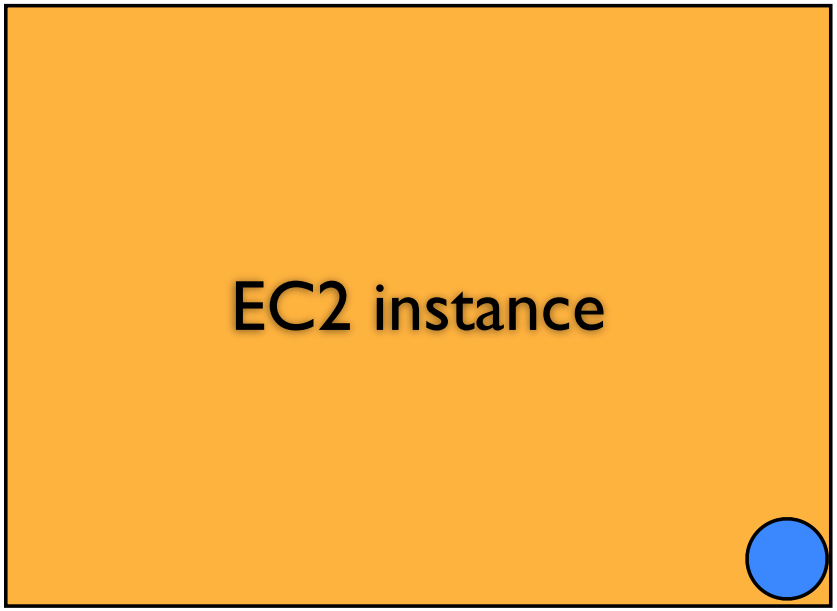
- An erlang process
- With LOCAL state
- responding to requests from clients

A unit of throughput!

Composing throughput with erlang



Composing throughput with erlang

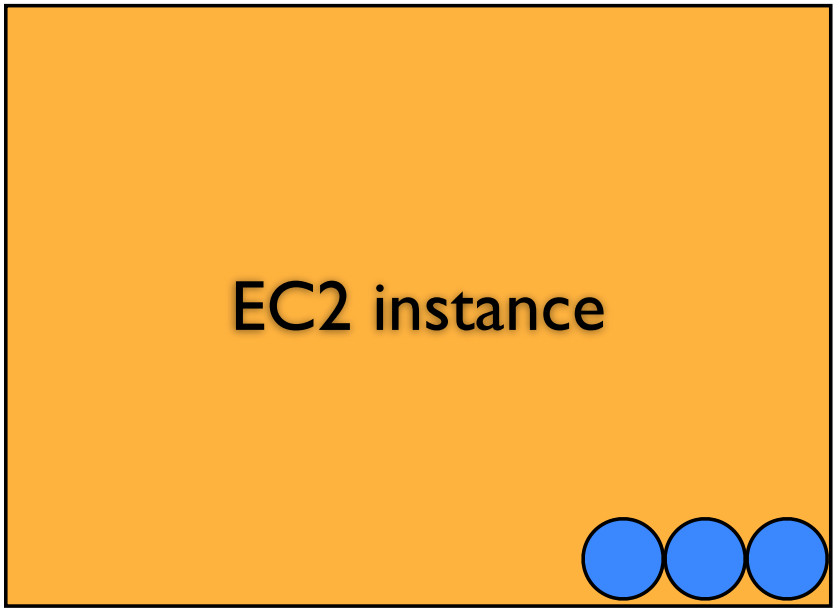


1 EC2 instance + 1 erlang VM

=

N kilo gen_servers (N kilo units of throughput)

Composing throughput with erlang

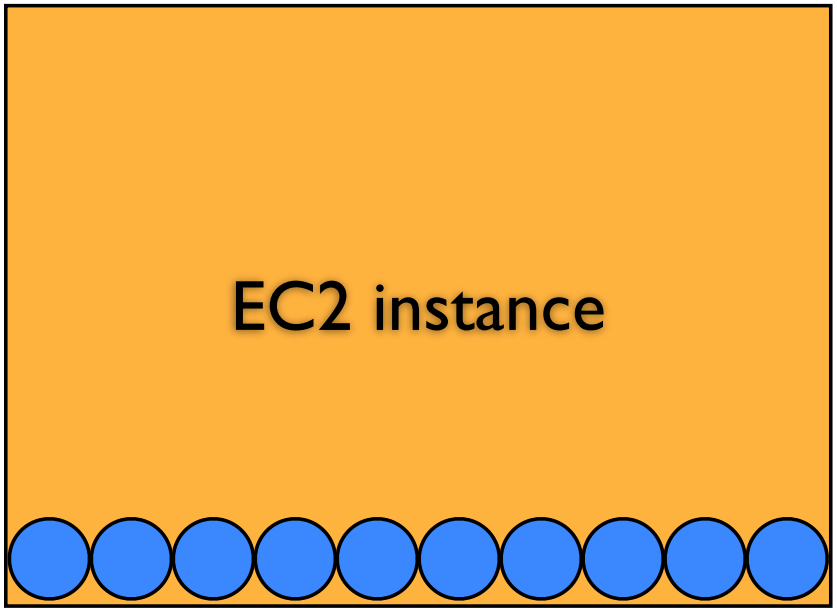


1 EC2 instance + 1 erlang VM

=

N kilo gen_servers (N kilo units of throughput)

Composing throughput with erlang

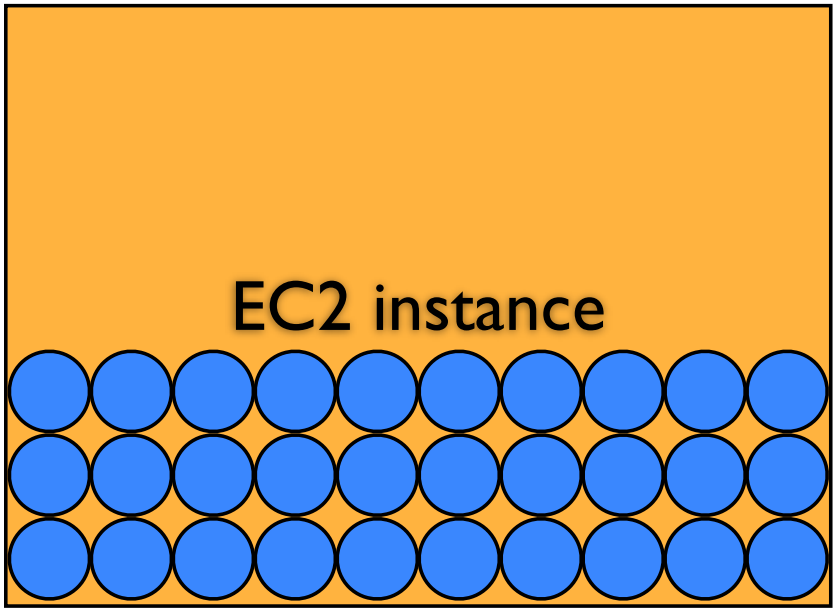


1 EC2 instance + 1 erlang VM

=

N kilo gen_servers (N kilo units of throughput)

Composing throughput with erlang

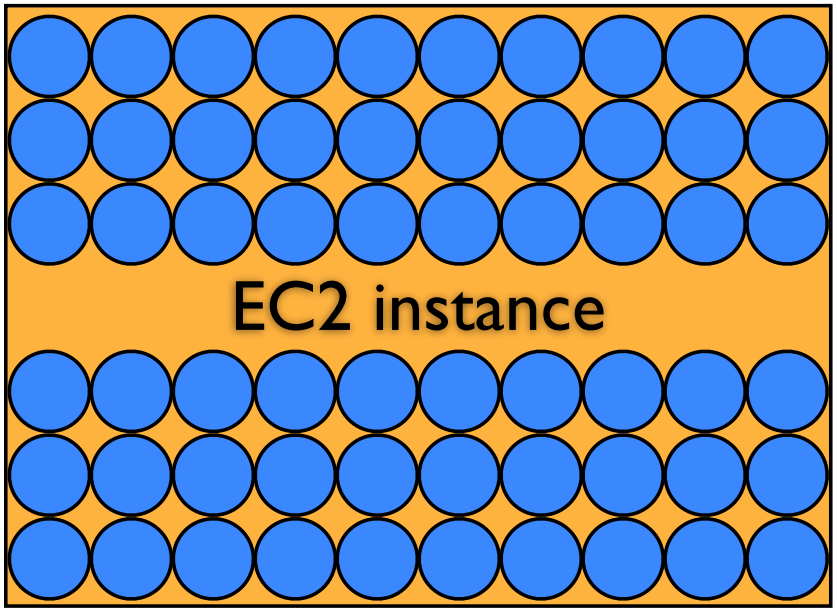


1 EC2 instance + 1 erlang VM

=

N kilo gen_servers (N kilo units of throughput)

Composing throughput with erlang

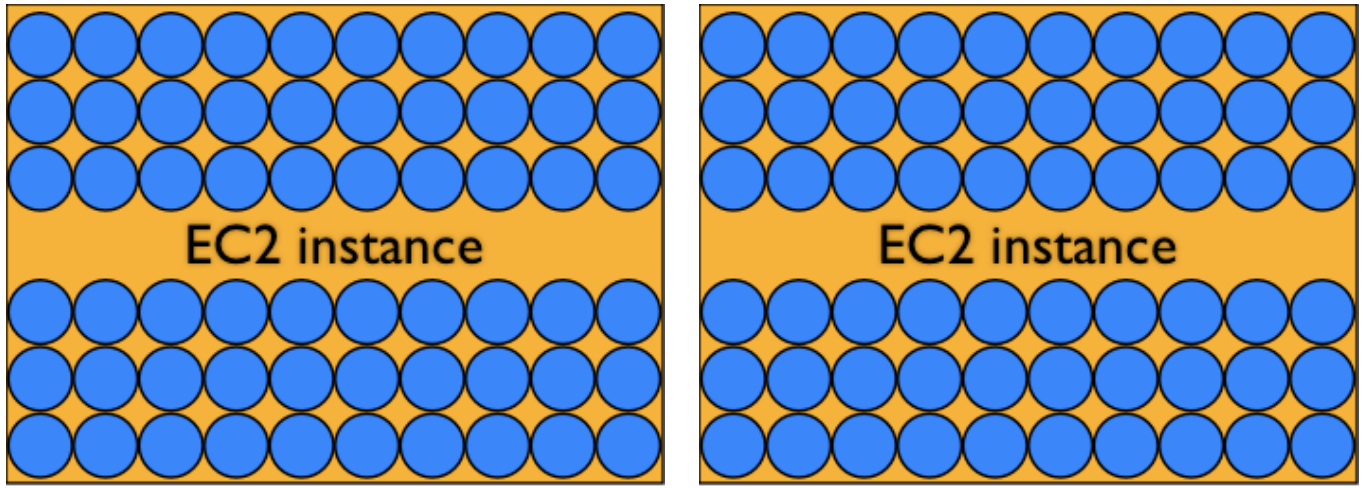


1 EC2 instance + 1 erlang VM

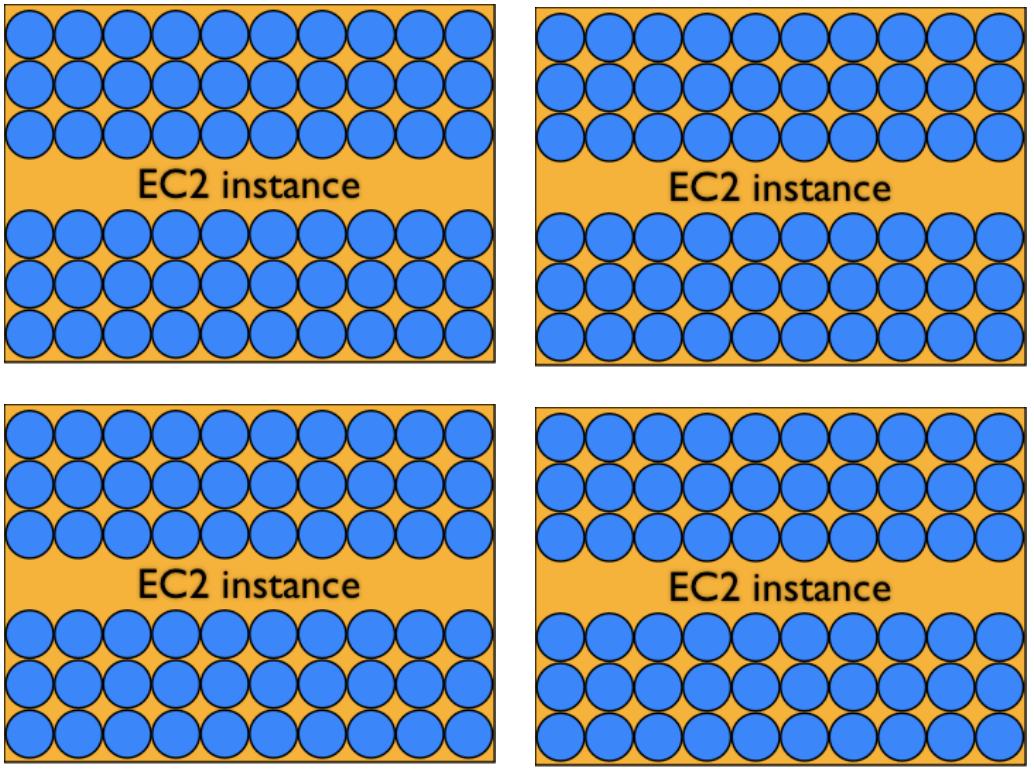
=

N kilo gen_servers (N kilo units of throughput)

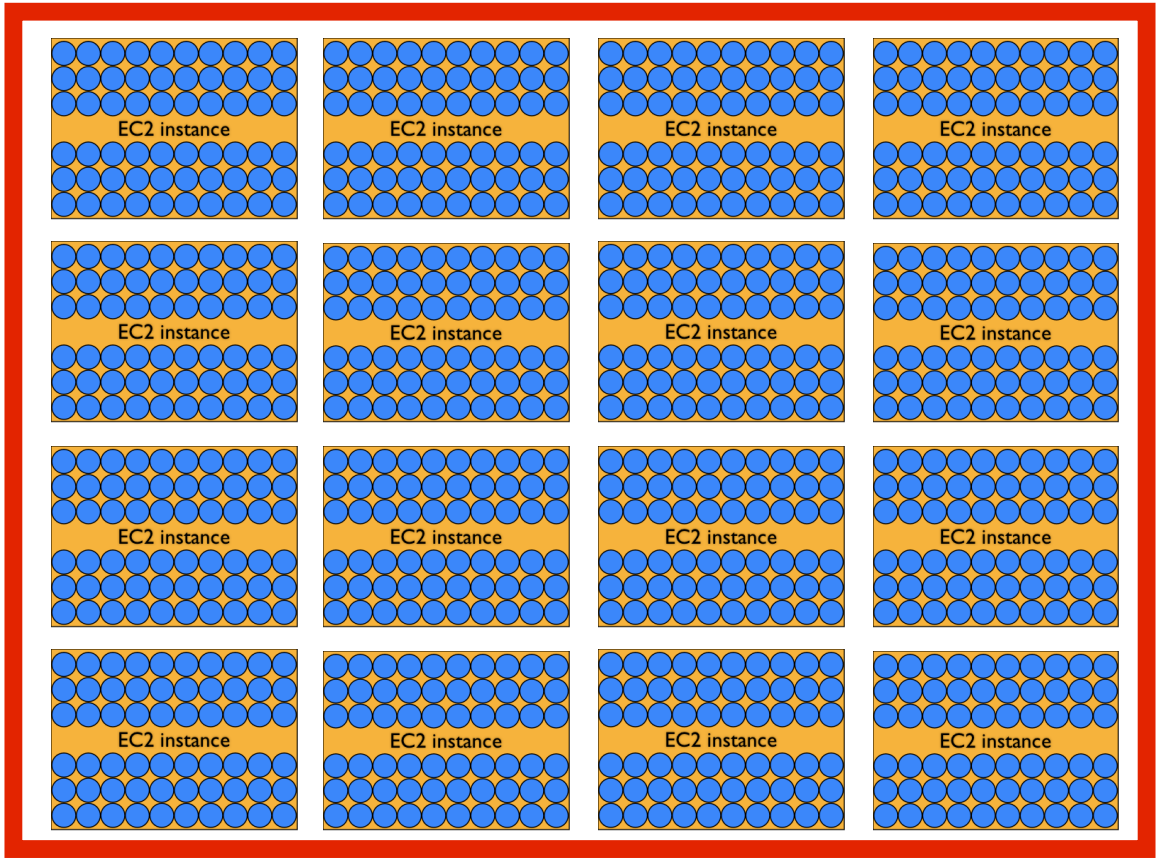
Scale by adding units instances



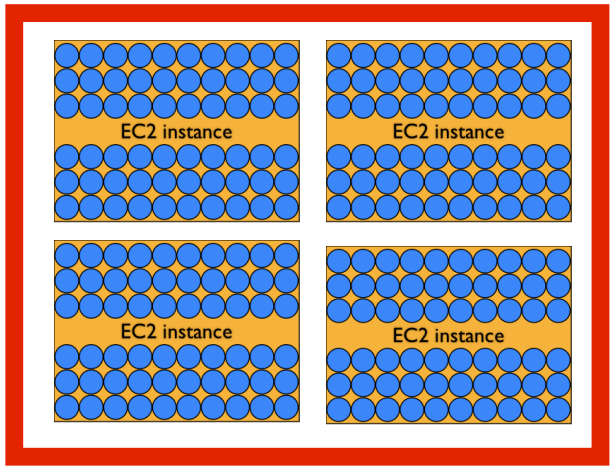
Scale up adding units instances



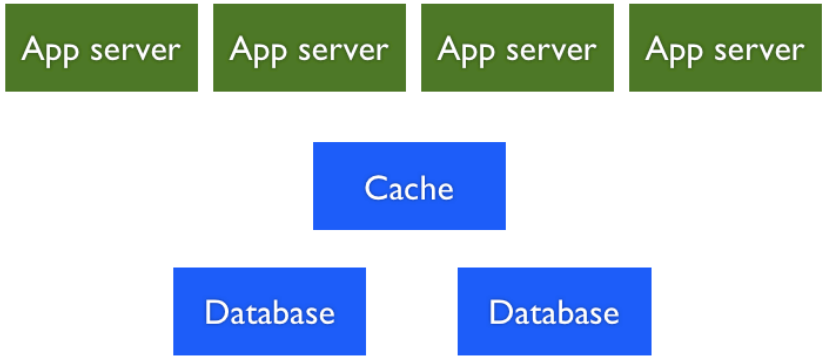
Erlang distribution



Throughput complexity



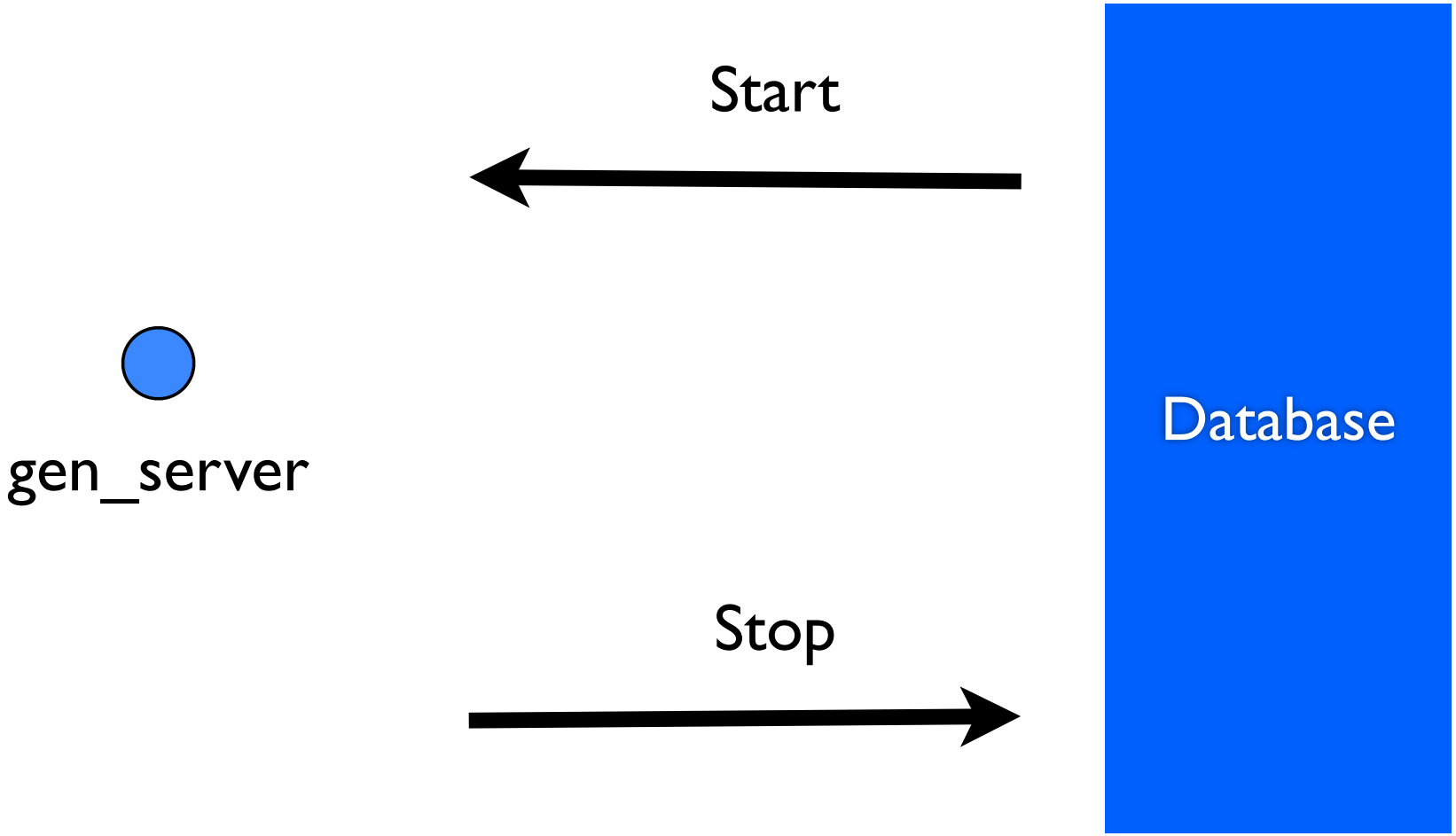
VS.



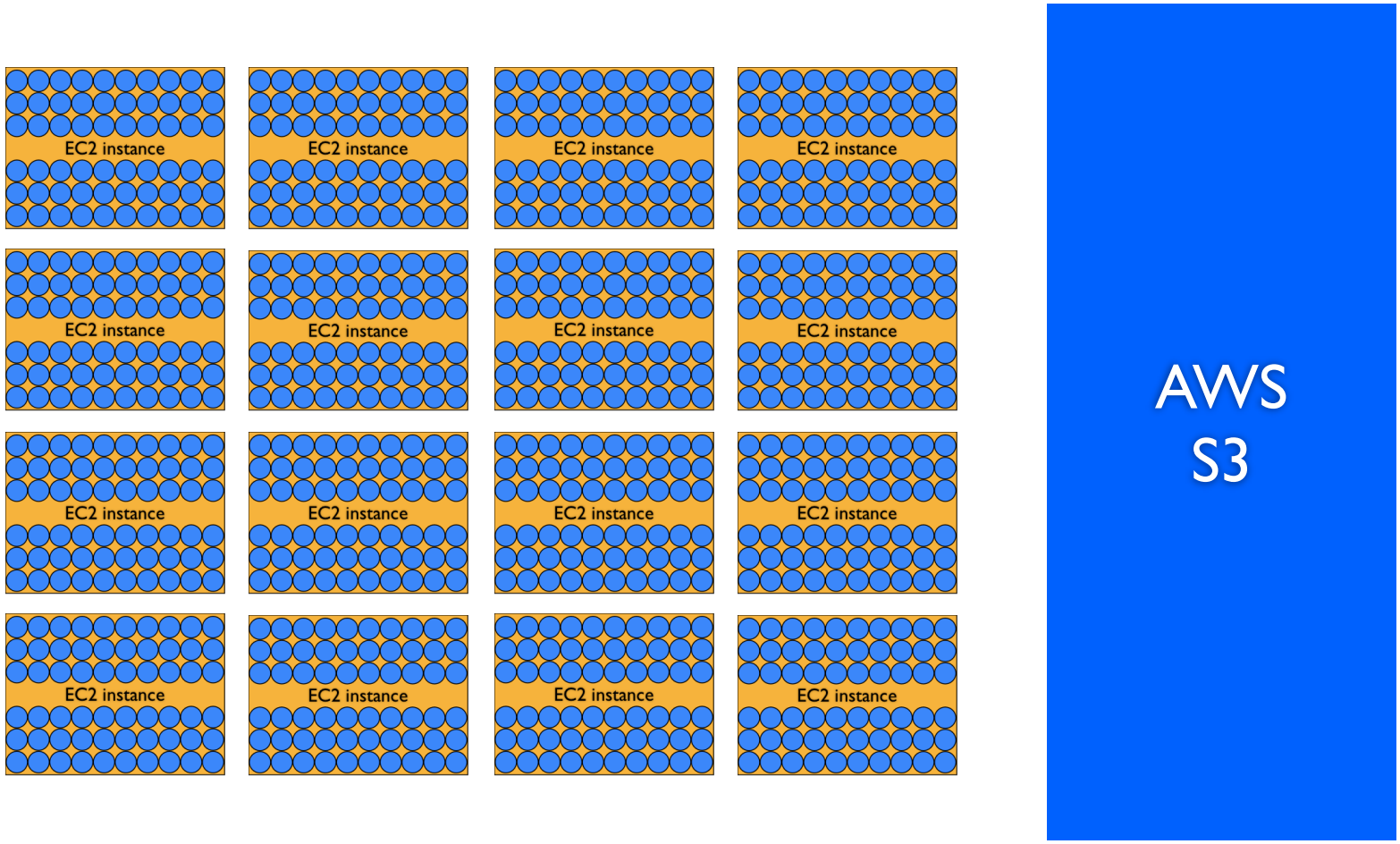
- Losely coupled peers
- Independent throughput

- Tightly coupled roles
- Dependent throughput

Where does the state come from?

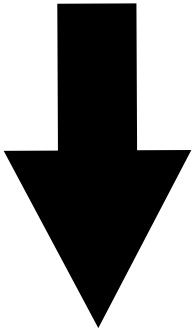


Now with database



Database scalability

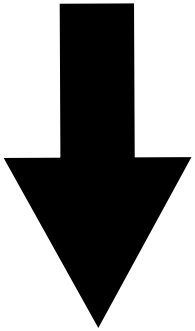
DB is (almost) never on
latency critical path



No need for low latency DB

Database scalability

Throughput required is low

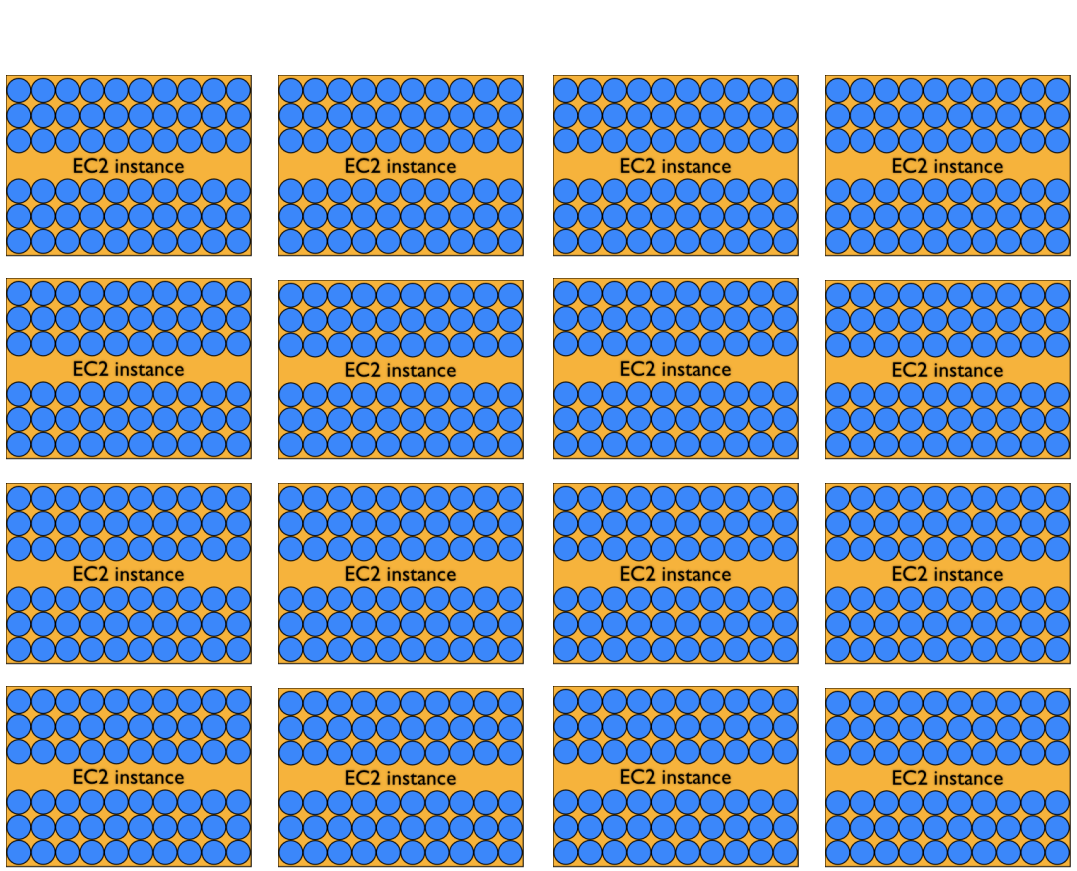


We can approximate S3 capacity as infinite

Database scalability

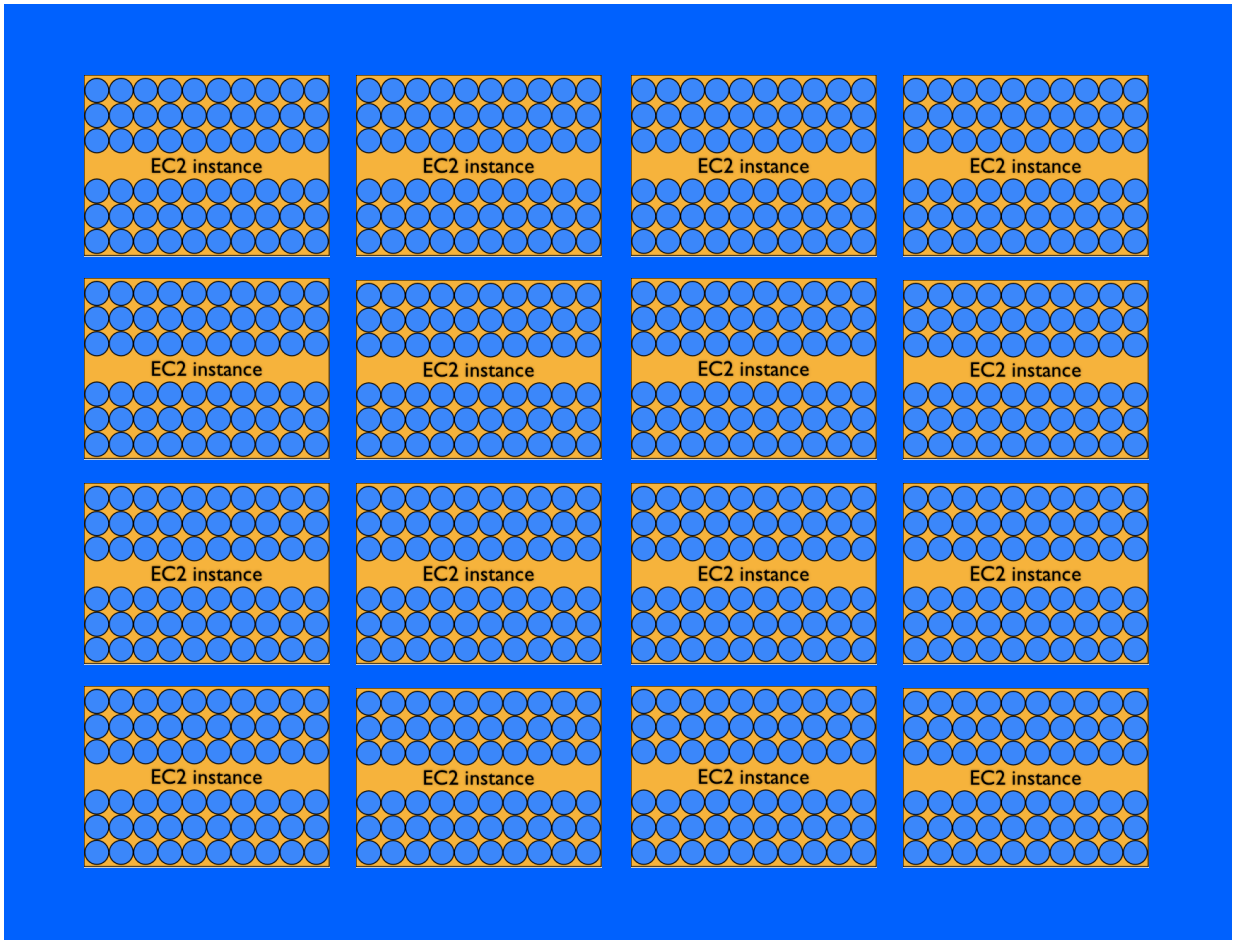
Ubiquitous and uniform from application servers point of view.

Remember?



AWS
S3

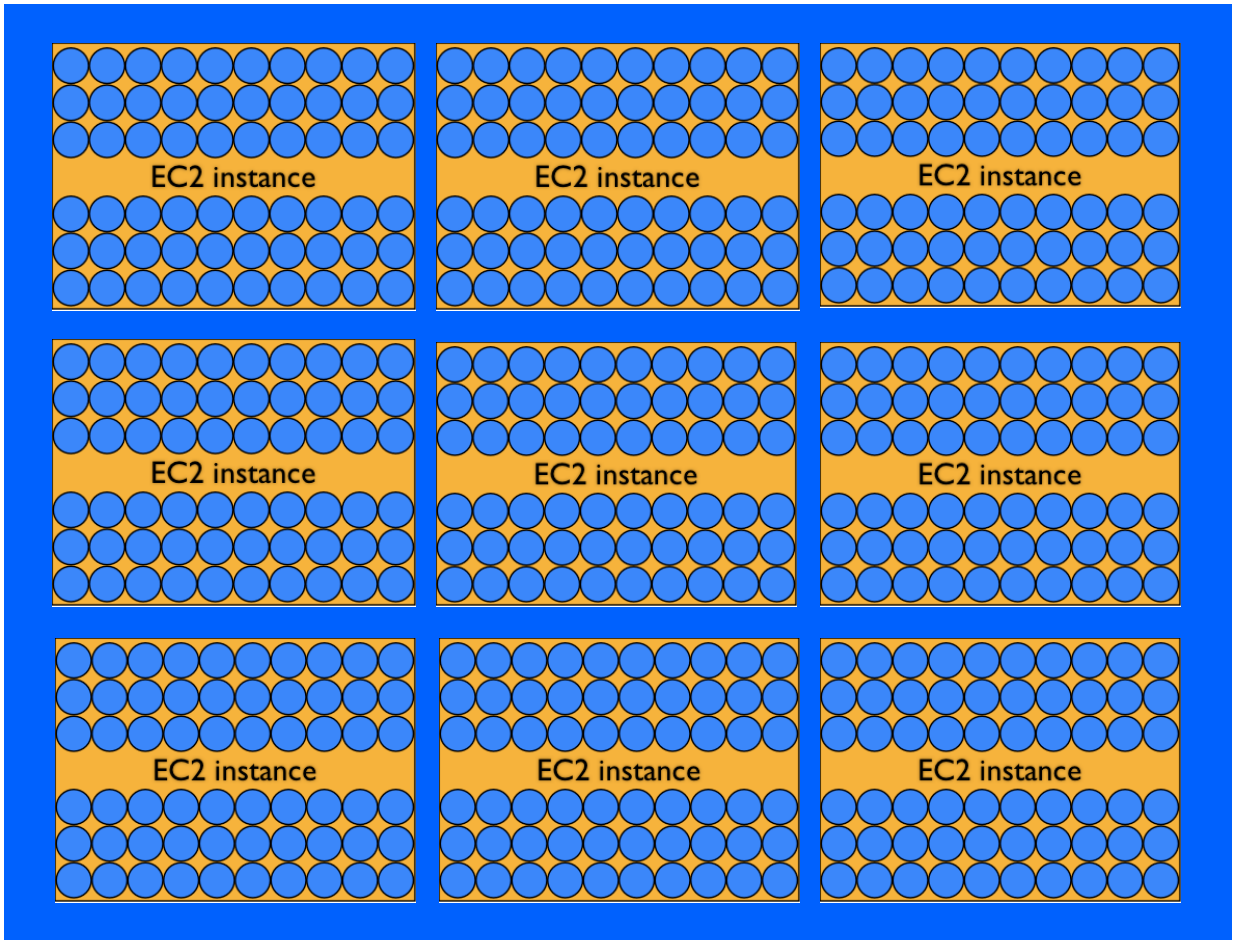
How it actually works



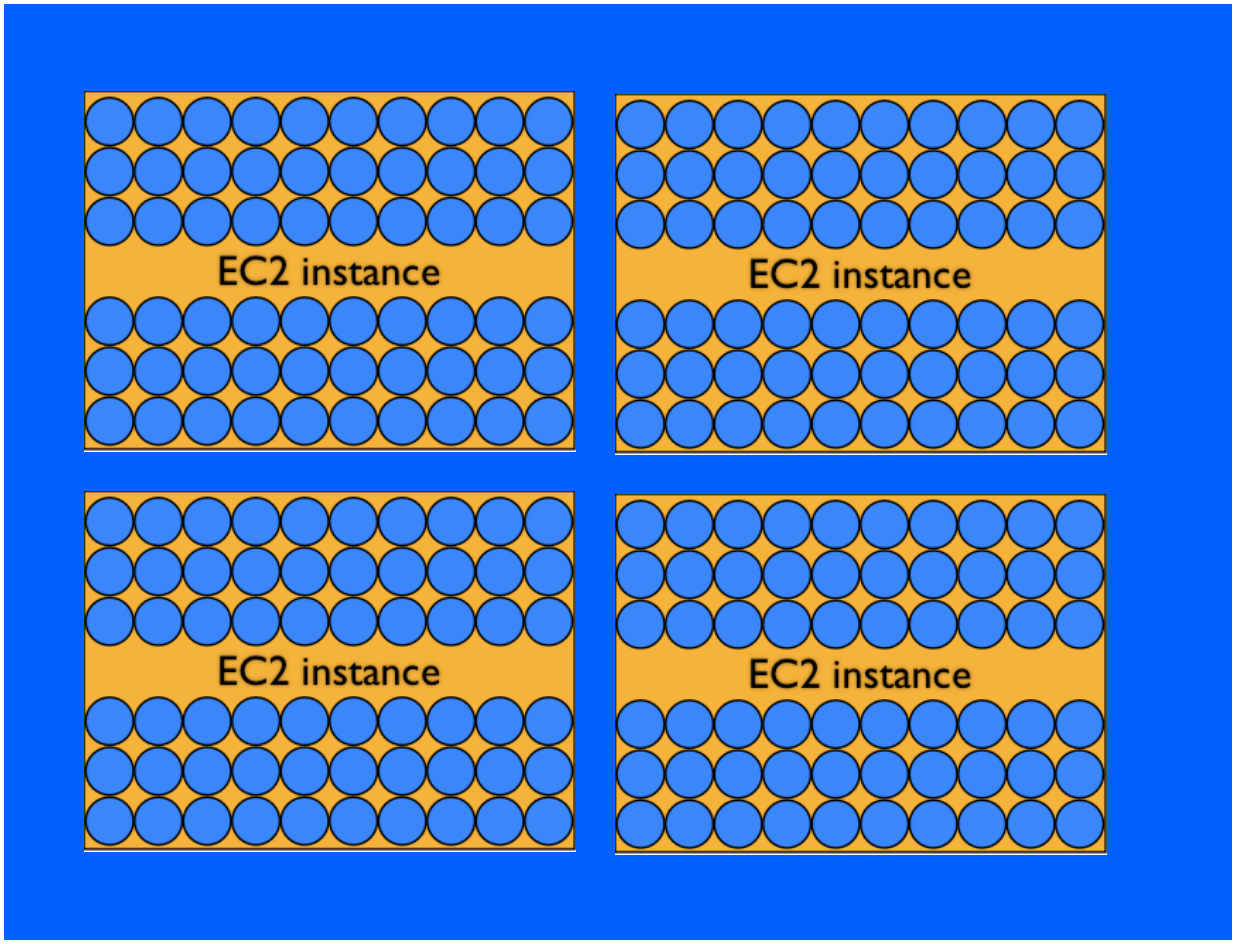
Data from the S3 is uniformly

available to any ec2 instance

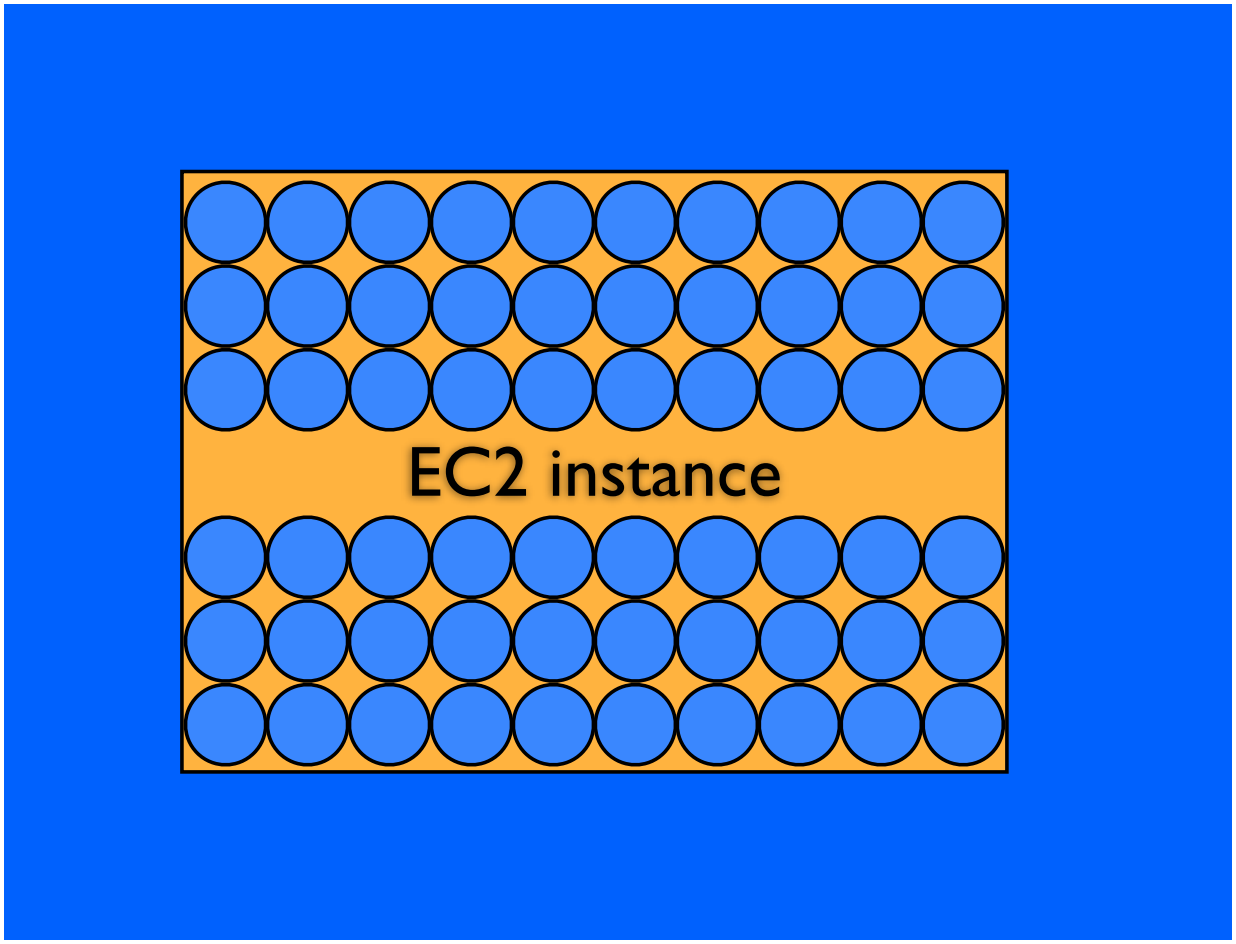
And as you zoom in...



And zoom in...

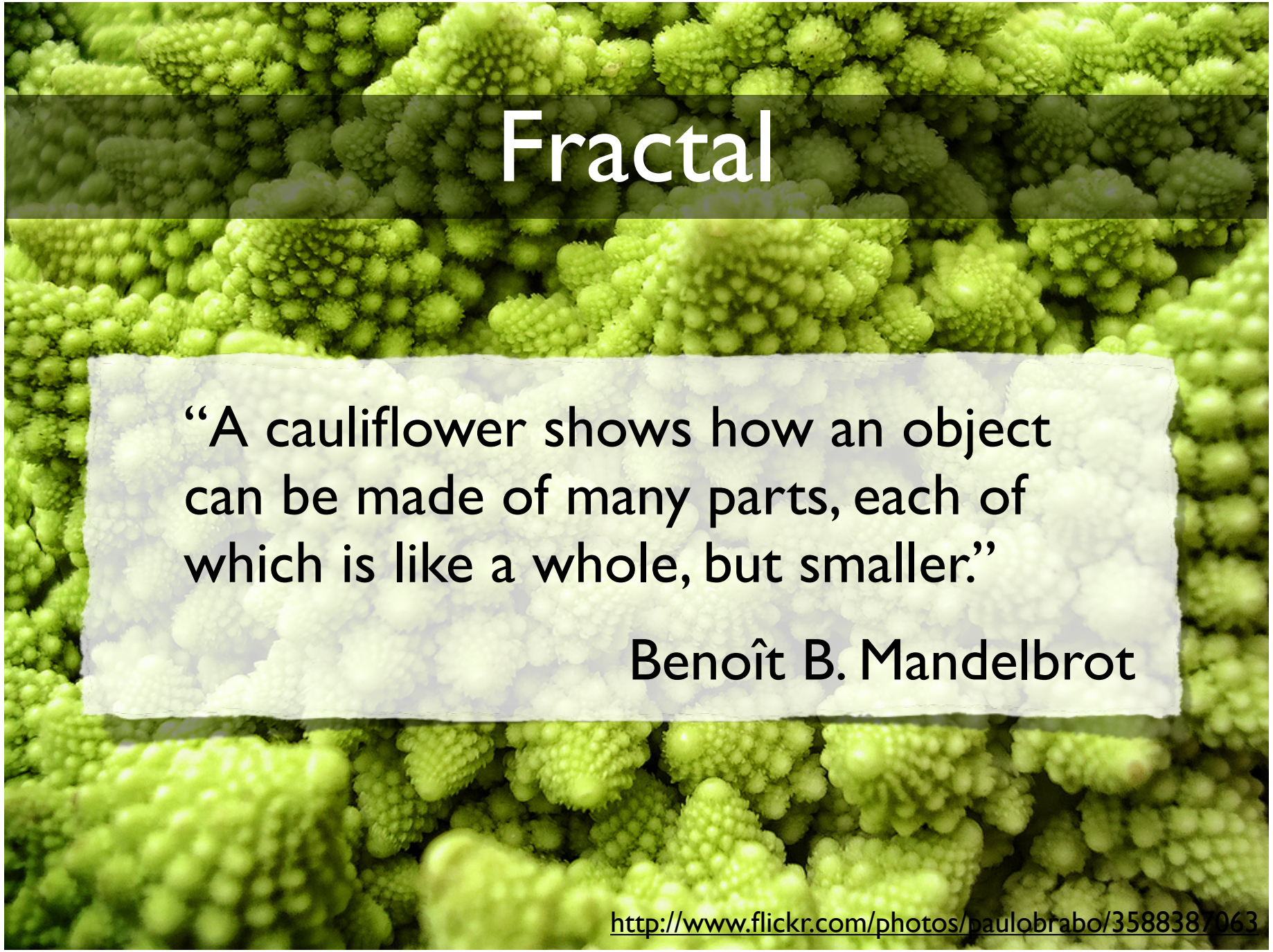


And zoom in you see...



Always the same
kind of structure

A fractal approach to
throughput


Fractal
"A cauliflower shows how an object can be made of many parts, each of which is like a whole, but smaller."
Benoît B. Mandelbrot
http://www.flickr.com/photos/paulobrabo/3588387063

Homework

The exact same solution might not work for you...

but look for that unit of throughput

Answer time

You need
XXXX
Smallish instances
to serve 0.25
billions uncacheable reqs/day

Answer time

You need
0XXX
Smallish instances
to serve 0.25
billions uncacheable reqs/day

Answer time

You need
00XX
Smallish instances
to serve 0.25
billions uncacheable reqs/day

Answer time

You need
0012
Smallish instances
to serve 0.25
billions uncacheable reqs/day

What's
1200 ?

Thanks

Paolo Negri @hungryblank

http://www.wooga.com/jobs

wooga

world of gaming