Autotuning and *Algorithmic Skeletons*

Uncontentious statement:

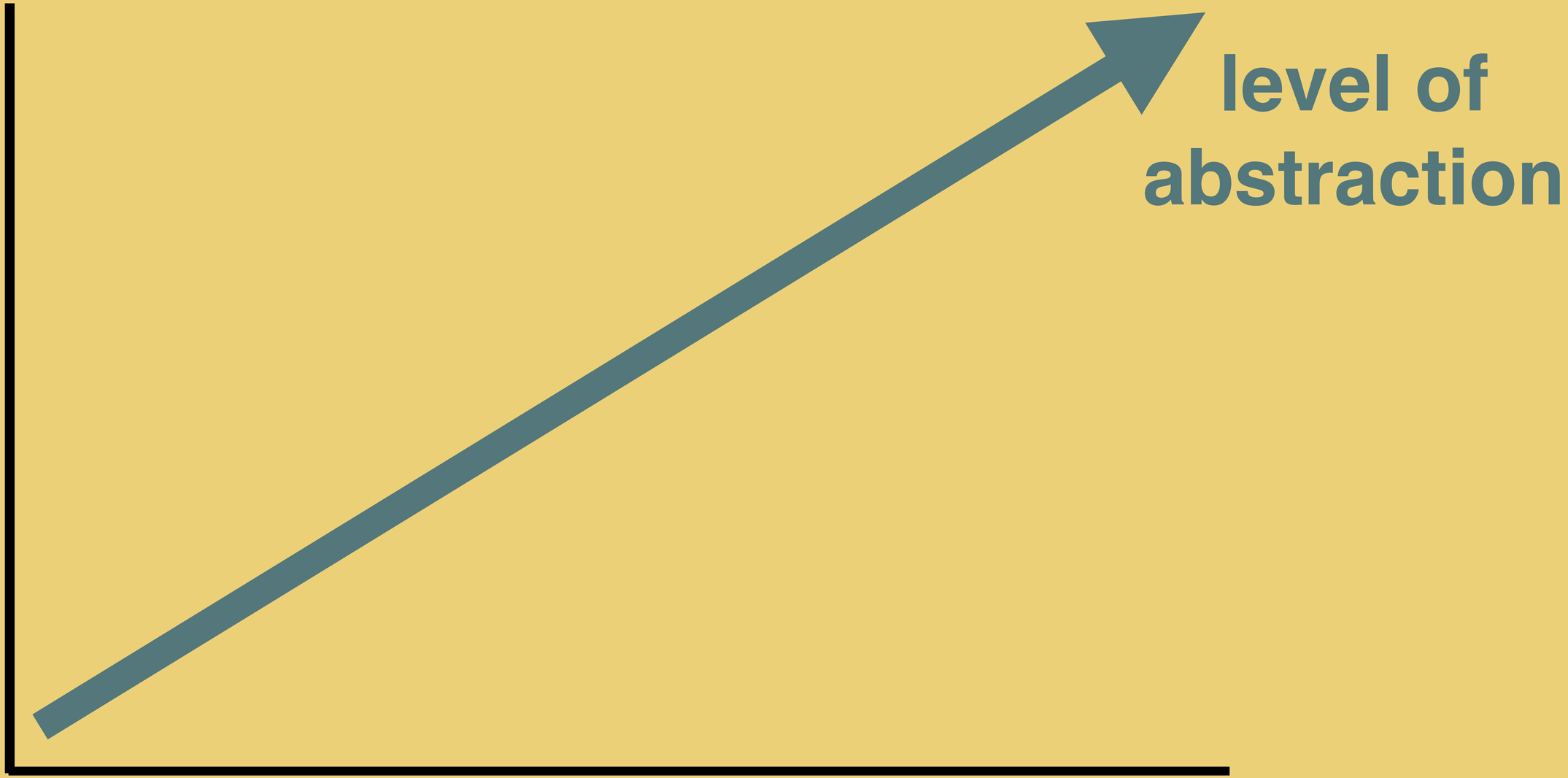
High level programming is great!

So why do application programmers resort to writing *low level* code?

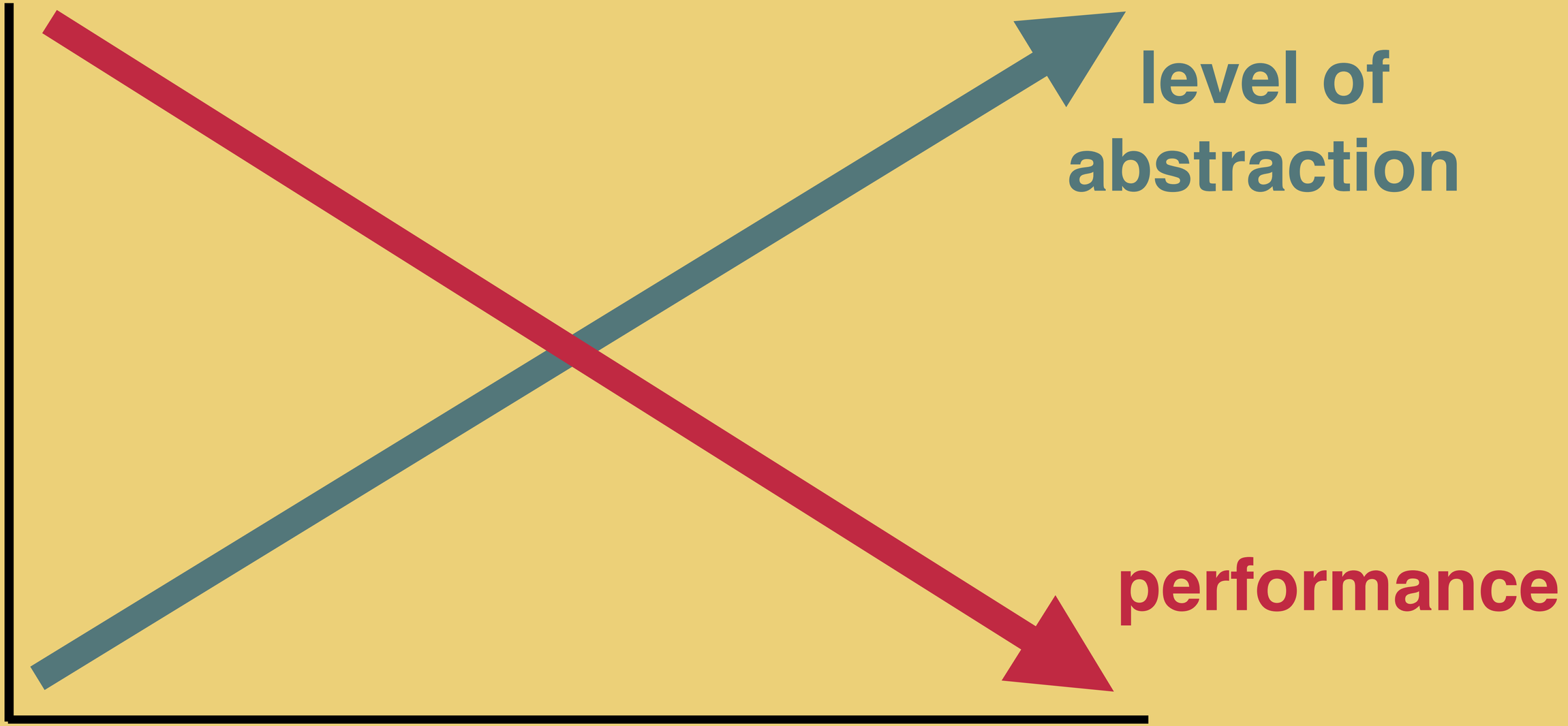
So why do application programmers resort to writing *low level* code?

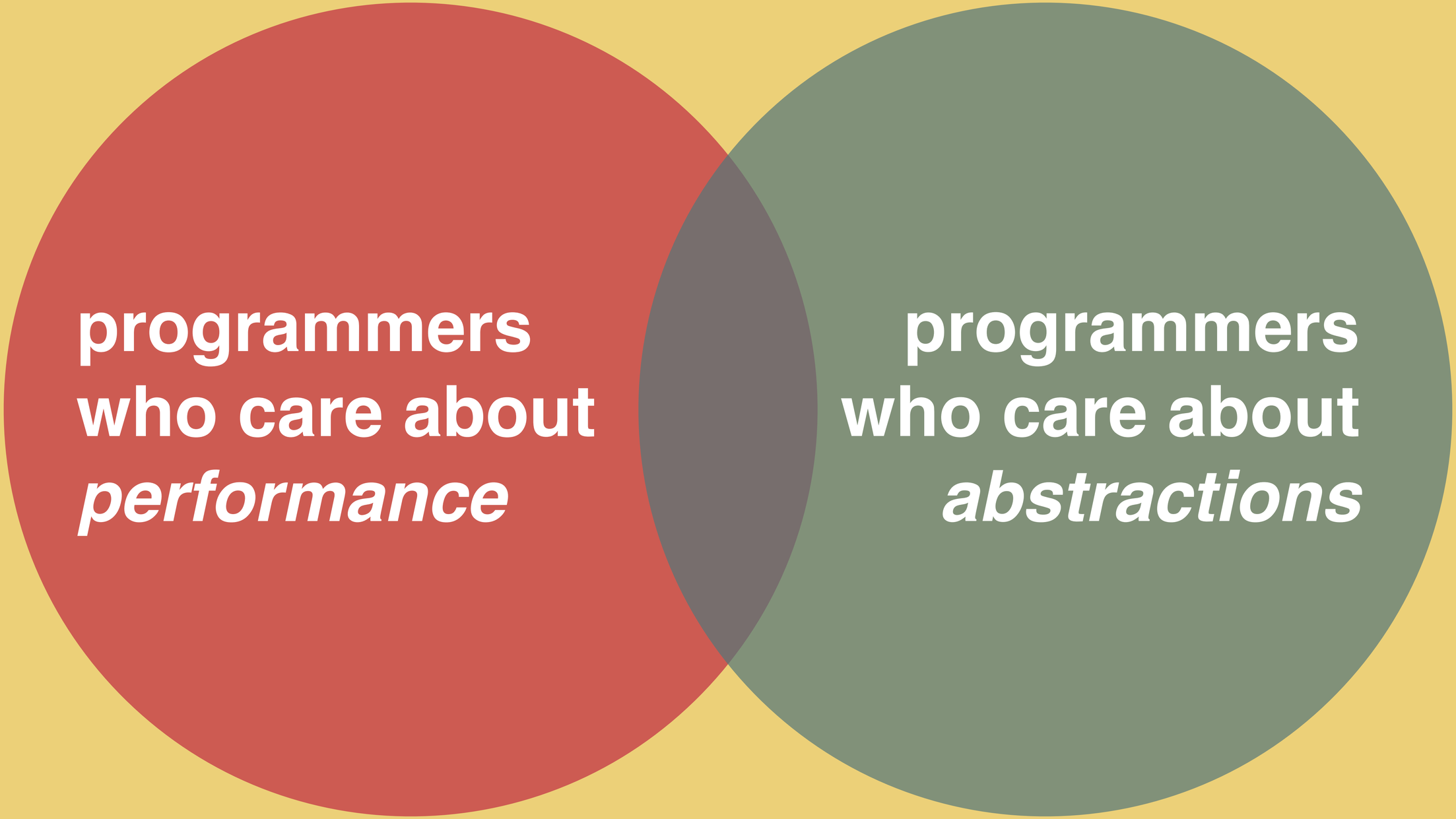
Performance!

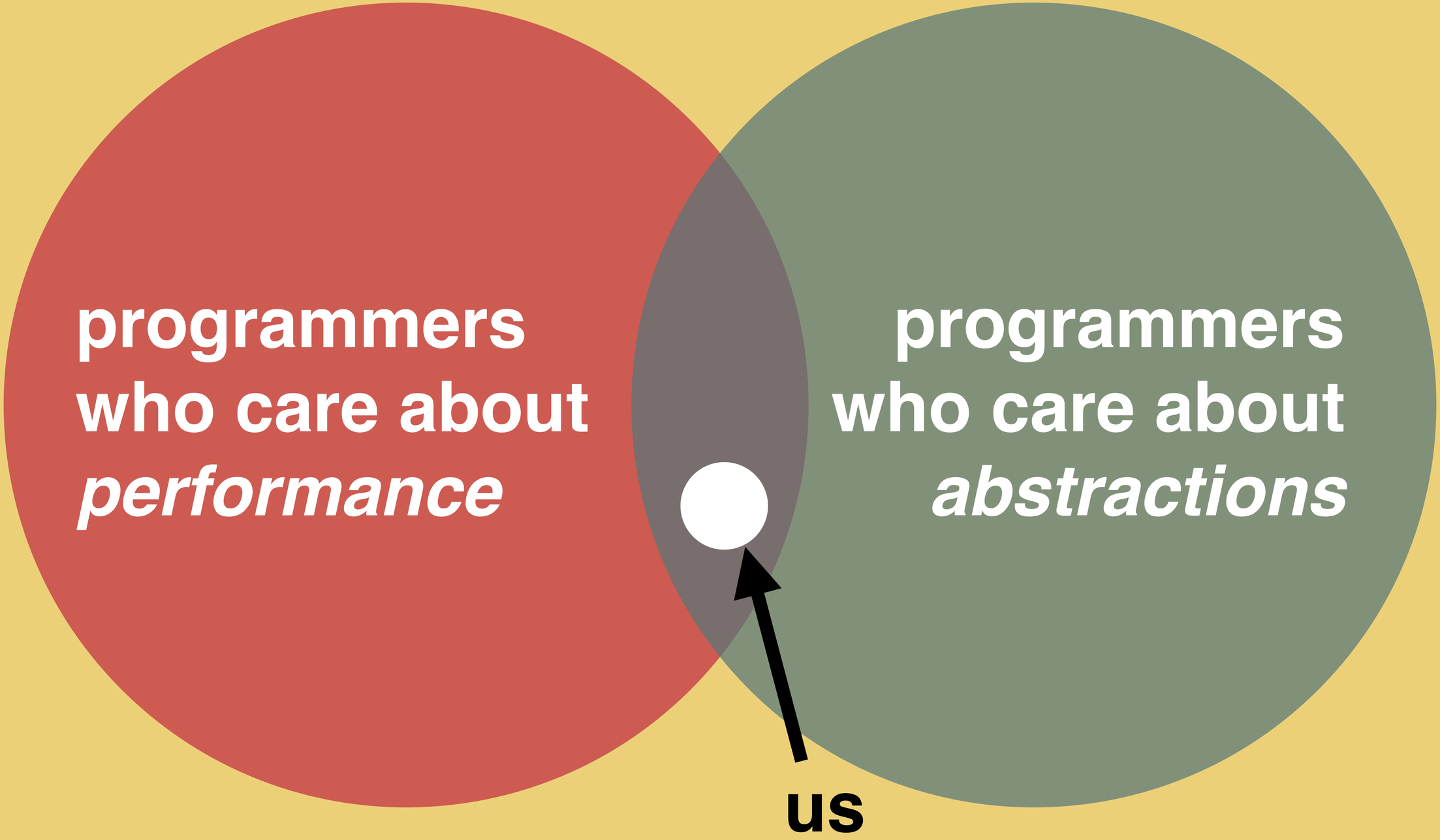
common perception is …

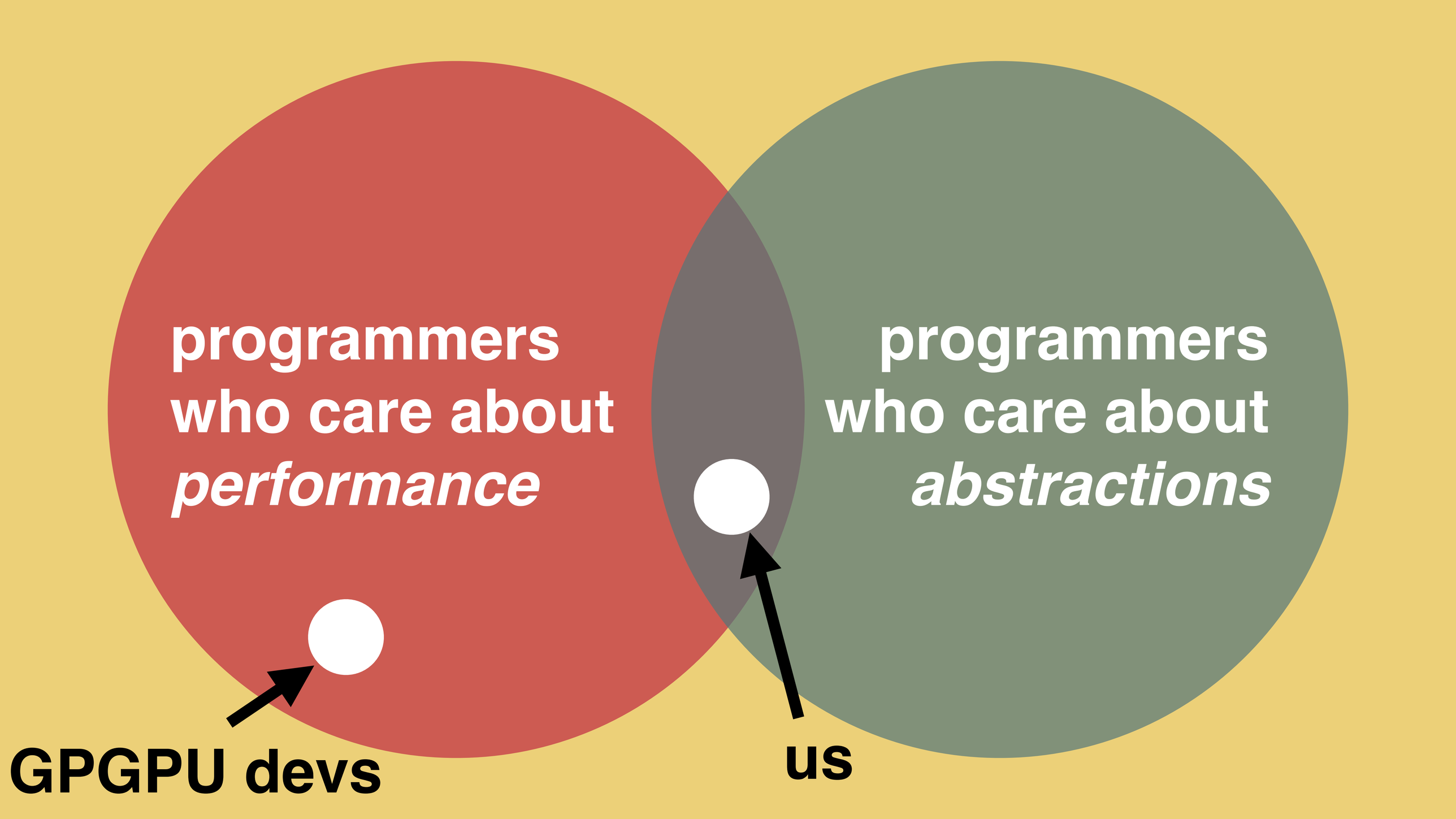


common perception is …




programmers
who care about
performance
programmers
who care about
abstractions


programmers
who care about
performance
programmers
who care about
abstractions
us


programmers
who care about
performance
programmers
who care about
abstractions
GPGPU devs
us

How do we break the illusion?

High level code needs to be at least *competitive* with low level

High level code needs to be at least *competitive* with low level (but faster would be nice)

Reasons for low level:

Reasons for low level: Domain-specific optimisations

Reasons for low level:

Domain-specific optimisations

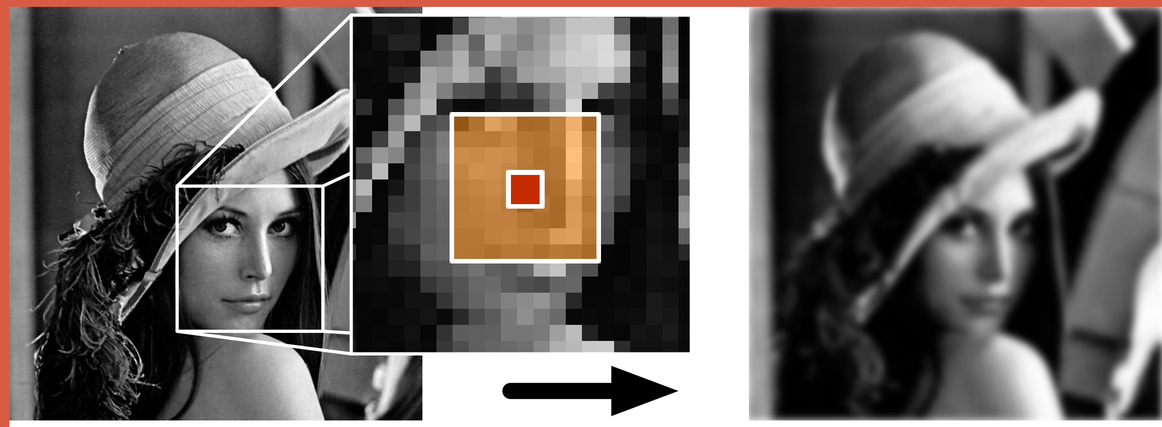
Parameter tuning

Reasons for low level: Domain-specific optimisations

Parameter tuning

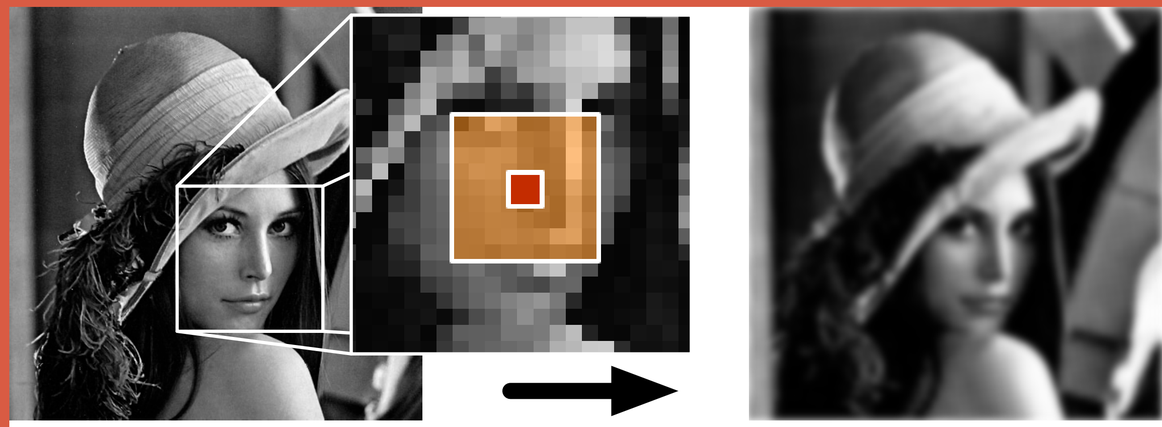
Parameter tuning for Algorithmic Skeletons

Parameter tuning for ~~Algorithmic Skeletons~~ stencils



OpenCL workgroup size

~~Parameter tuning~~ for ~~Algorithmic Skeletons~~ stencils



OpenCL workgroup size:

OpenCL workgroup size:

Controls composition of hardware threads.

OpenCL workgroup size:

Controls composition of hardware threads.

Is a 2D parameter (rows x cols).

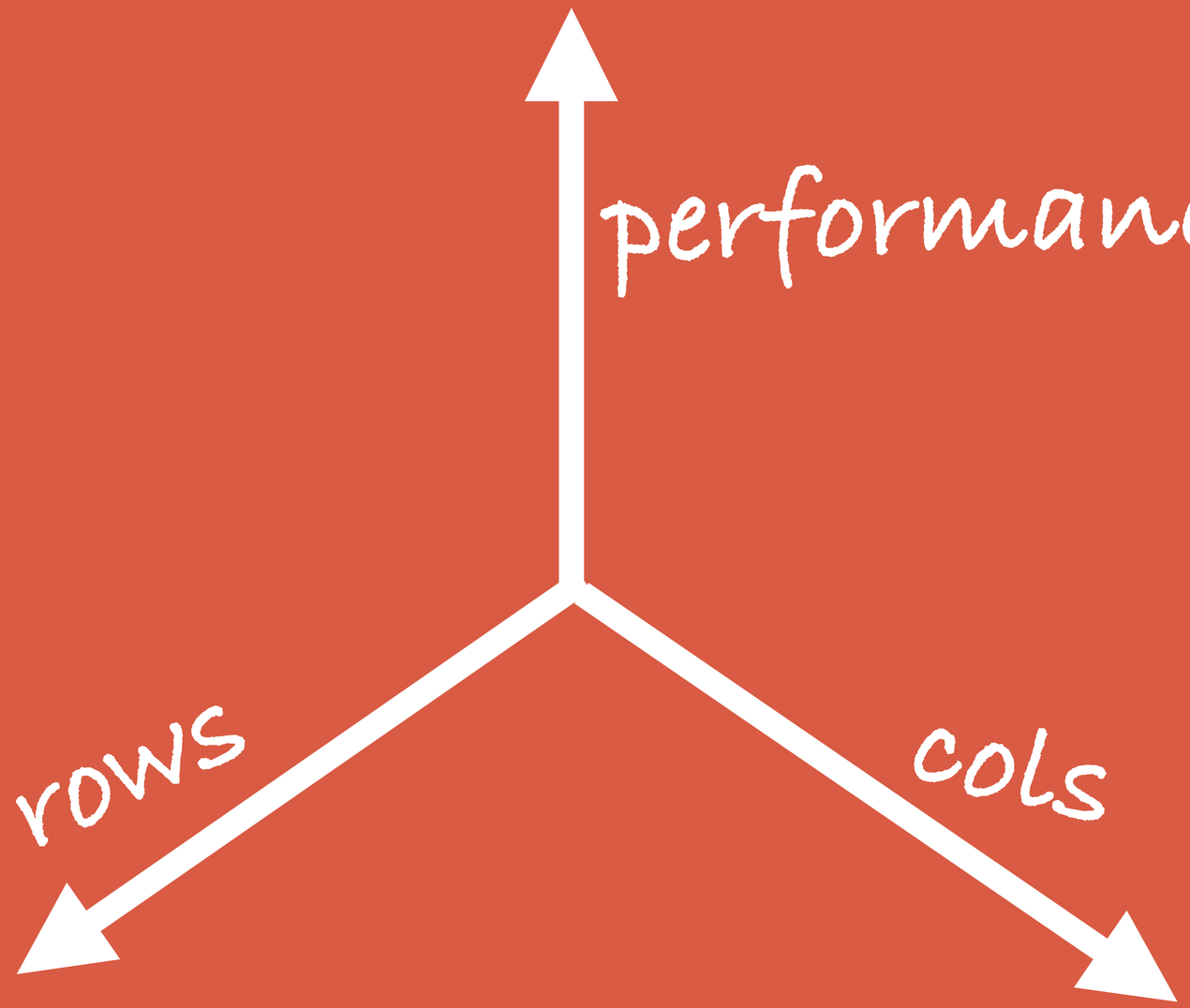
OpenCL workgroup size:

Controls composition of hardware threads.

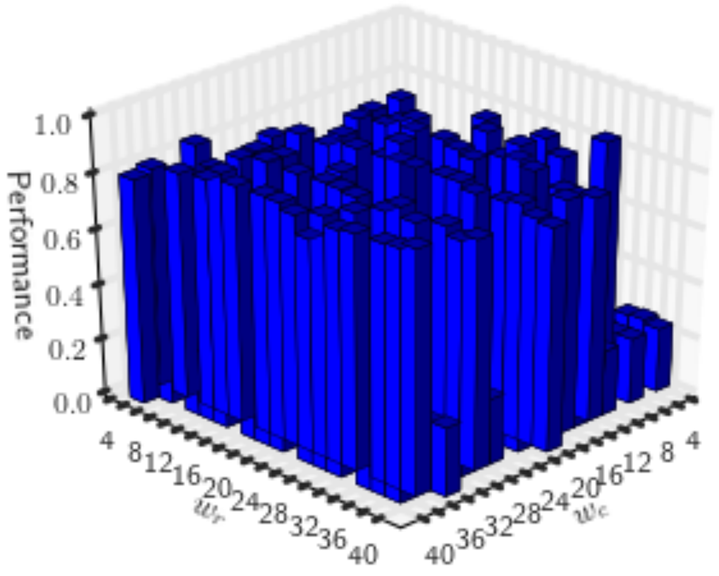
Is a 2D parameter (rows x cols).

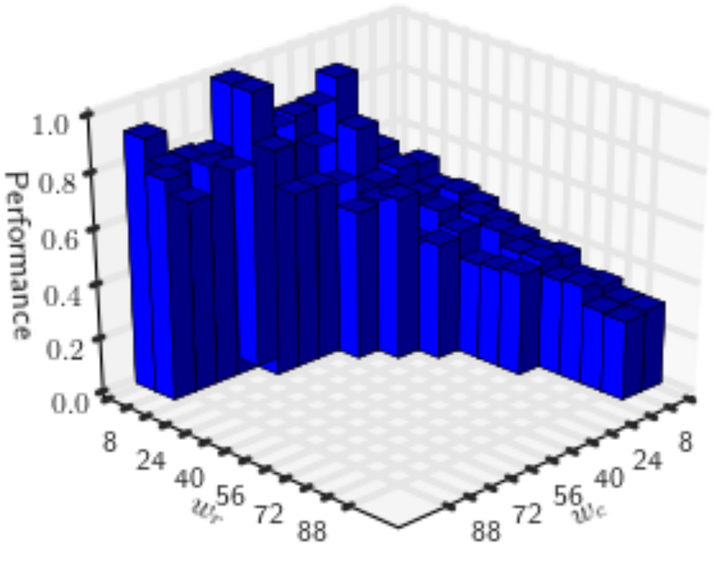
Critical to performance.

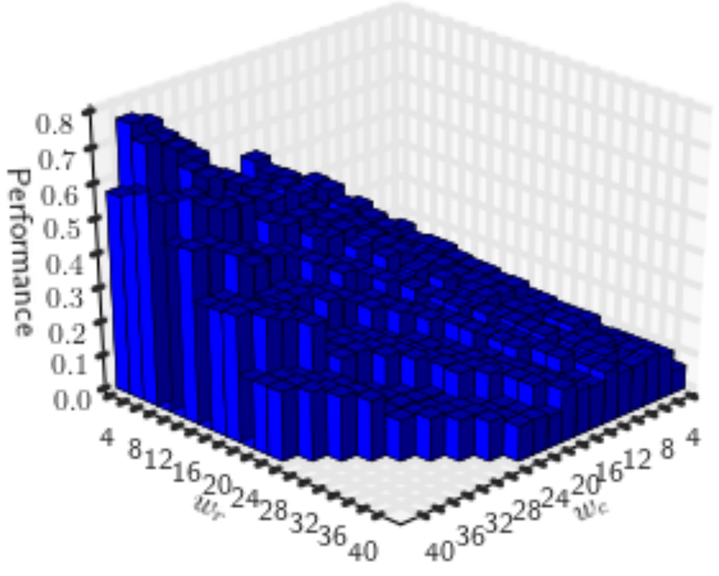
OpenCL workgroup size:

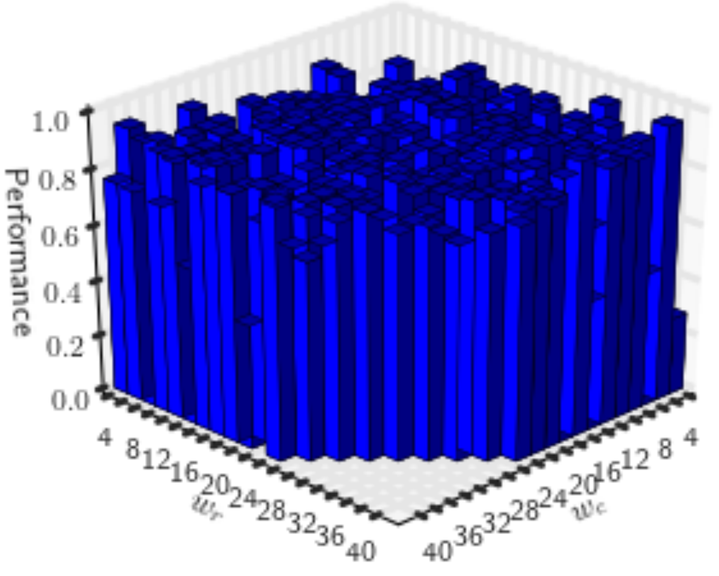


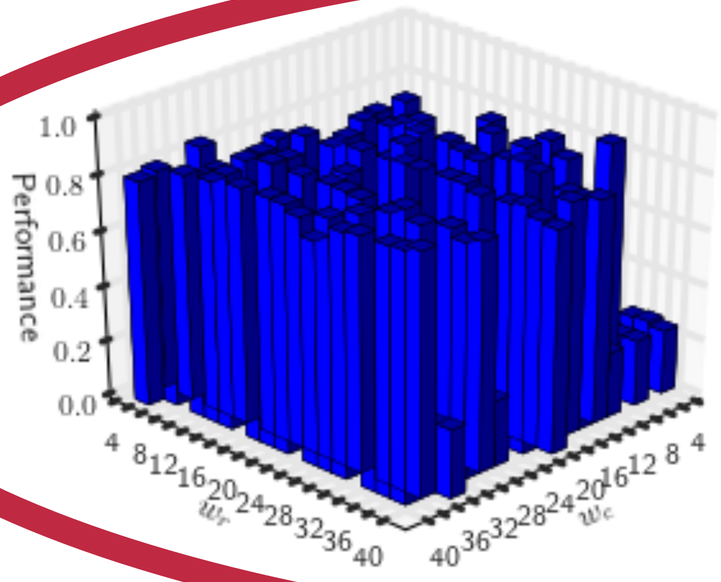
Examples

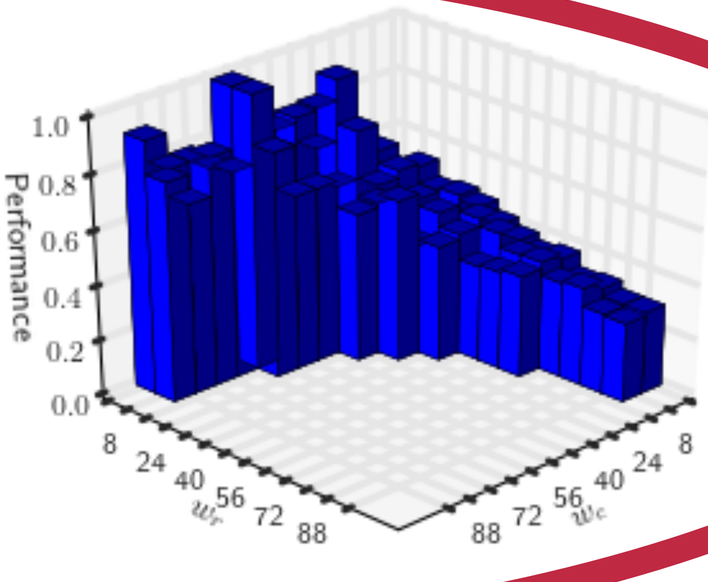

Performance
1.0
0.8
0.6
0.4
0.2
0.0
4 8 12 16 20 24 28 32 36 40
w_r
40 36 32 28 24 20 16 12 8 4
w_c


Performance
1.0
0.8
0.6
0.4
0.2
0.0
8 24 40 56 72 88
w_r
88 72 56 40 24 8
w_c


Performance
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
4 8 12 16 20 24 28 32 36 40
w_r
40 36 32 28 24 20 16 12 8 4
w_c

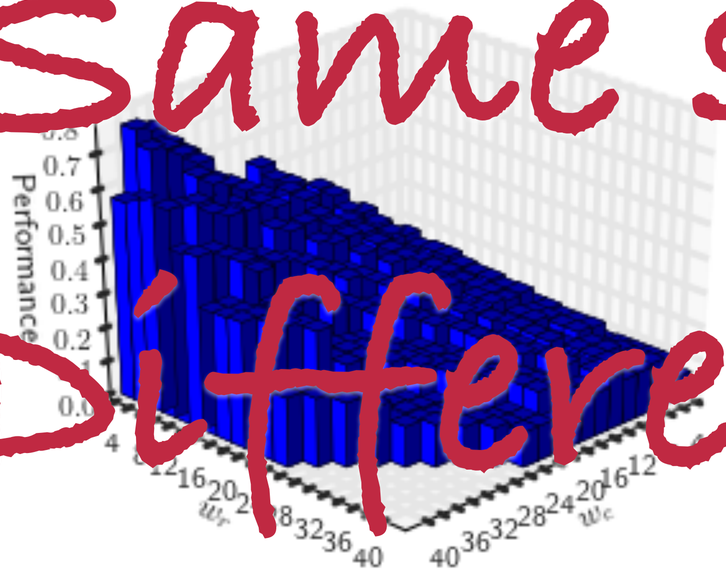

Performance
1.0
0.8
0.6
0.4
0.2
0.0
4 8 12 16 20 24 28 32 36 40
w_r
40 36 32 28 24 20 16 12 8 4
w_c

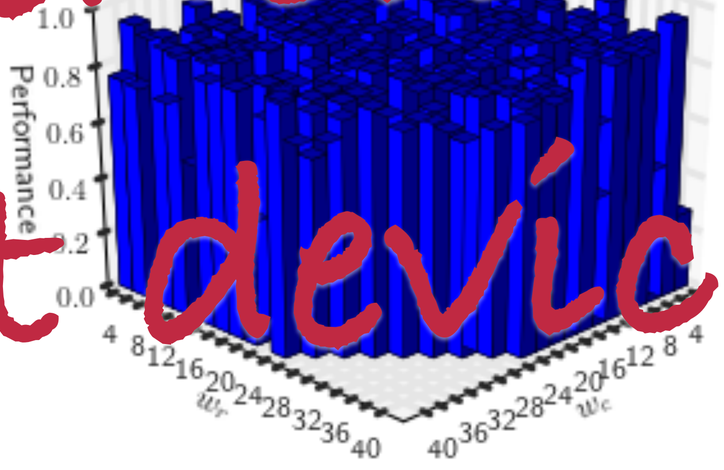




Same stencil!

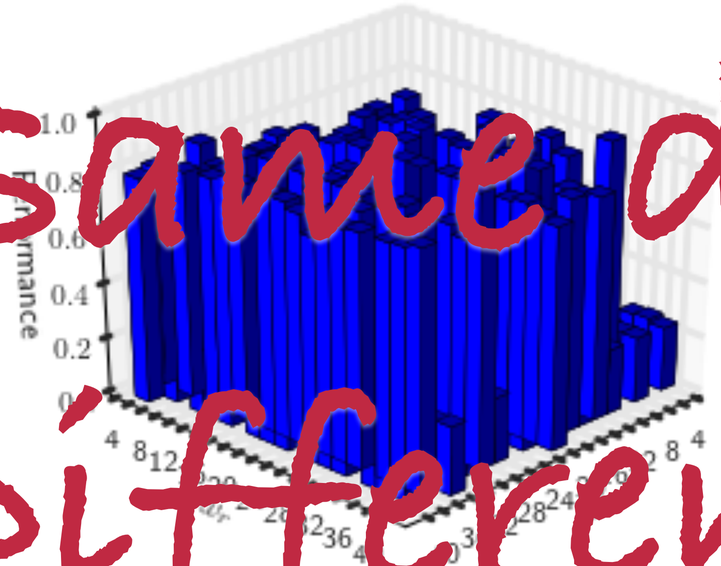
Different device!

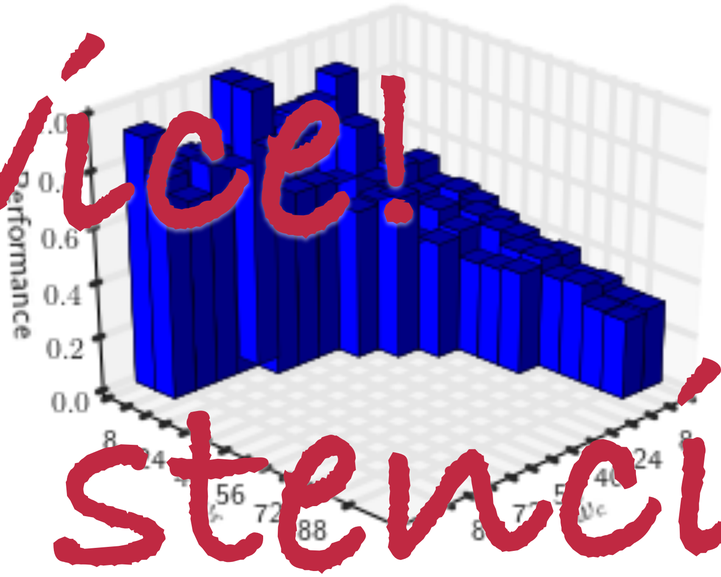


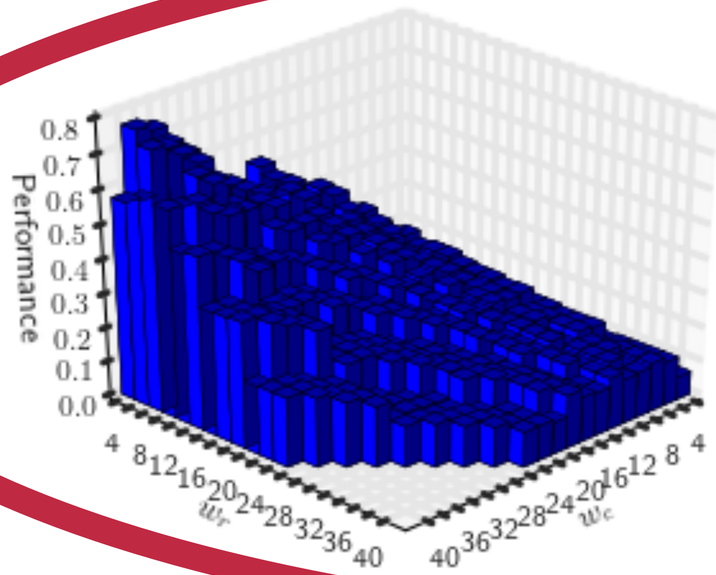


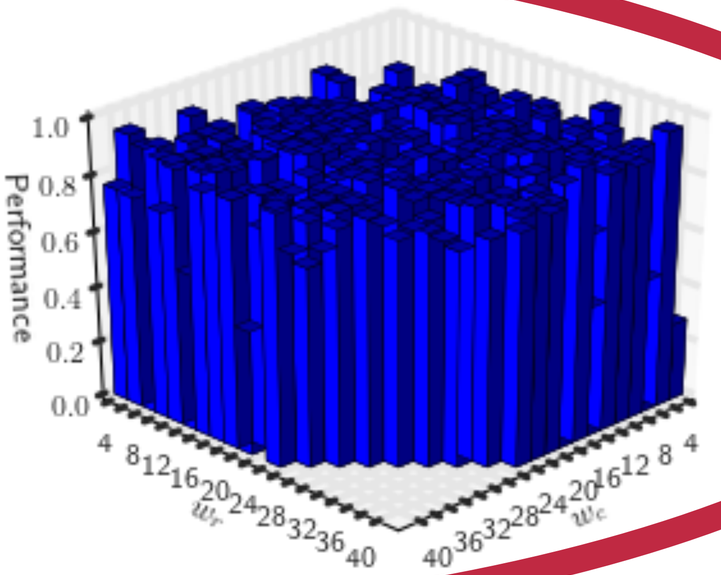
Same device!

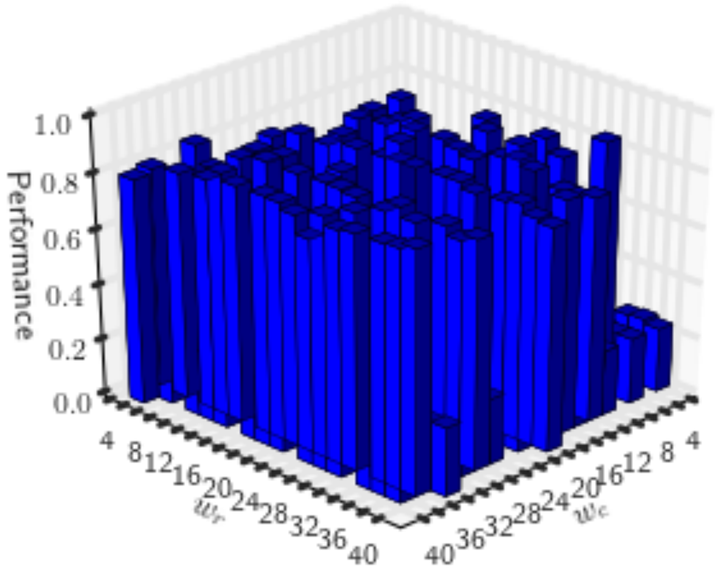
Different stencil!

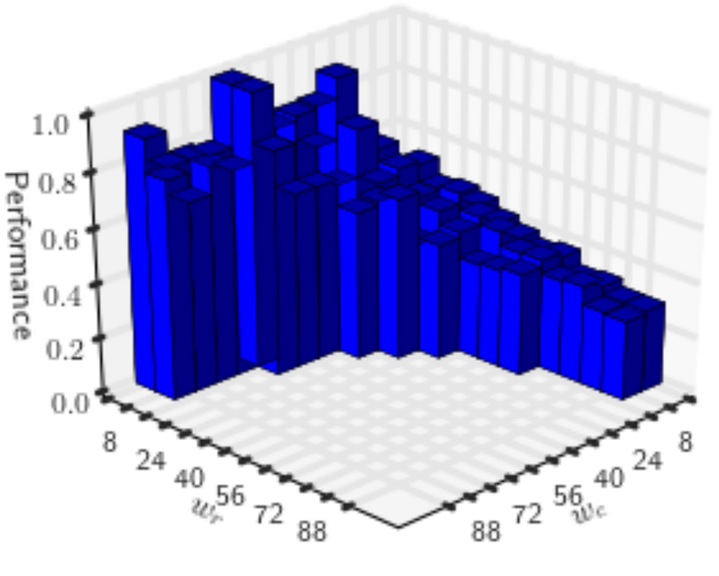


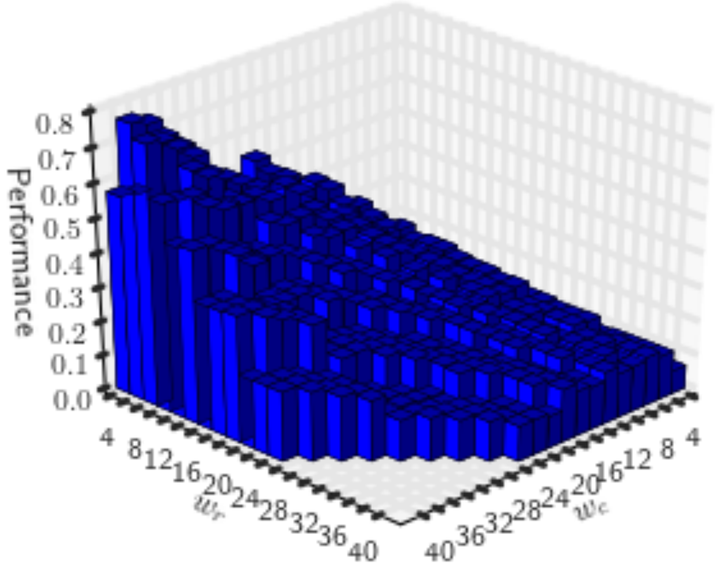


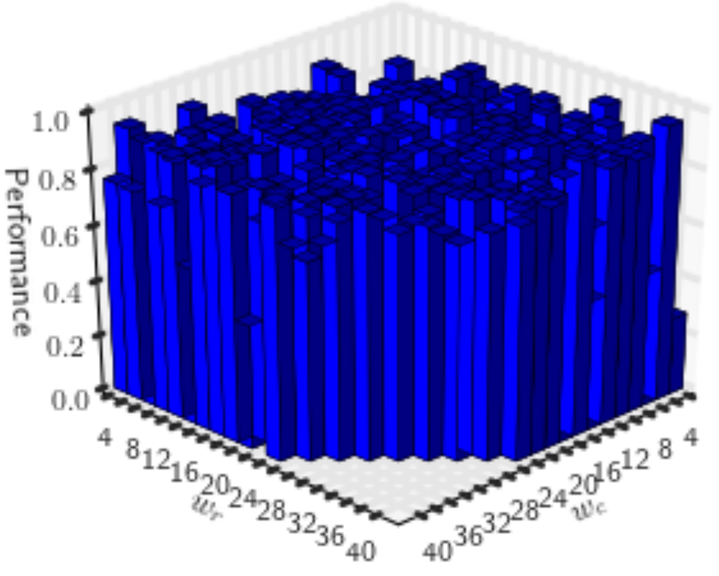





Performance
1.0
0.8
0.6
0.4
0.2
0.0
4 8 12 16 20 24 28 32 36 40
w_r
40 36 32 28 24 20 16 12 8 4
w_c


Performance
1.0
0.8
0.6
0.4
0.2
0.0
8 24 40 56 72 88
w_r
88 72 56 40 24 8
w_c


Performance
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
4 8 12 16 20 24 28 32 36 40
w_r
40 36 32 28 24 20 16 12 8 4
w_c


Performance
1.0
0.8
0.6
0.4
0.2
0.0
4 8 12 16 20 24 28 32 36 40
w_r
40 36 32 28 24 20 16 12 8 4
w_c

Choosing workgroup size depends on:

1. device
2. program
3. dataset

Picking good parameters

Approach 1

Set a workgroup size
Execute and time program

Set a workgroup size
Execute and time program

Set a workgroup size
Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

… (continue until done / bored)

Pick the best one you tried

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

… (continue until done / bored)

Pick the best one you tried

a job for the interns

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

… (continue until done / bored)

Pick the best one you tried

a job for the interns / iterative compilation

BAD!

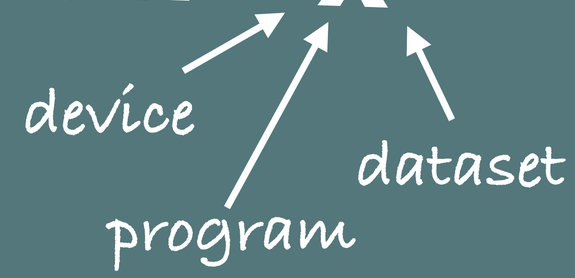
Takes a loooong time

BAD!

Takes a loooong time

BAD!

Must be repeated for every new “x”



Approach 2

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

… (continue until done / bored)

Pick the best one you tried

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

Set a workgroup size

Execute and time program

1 data point

… (continue until done / bored)

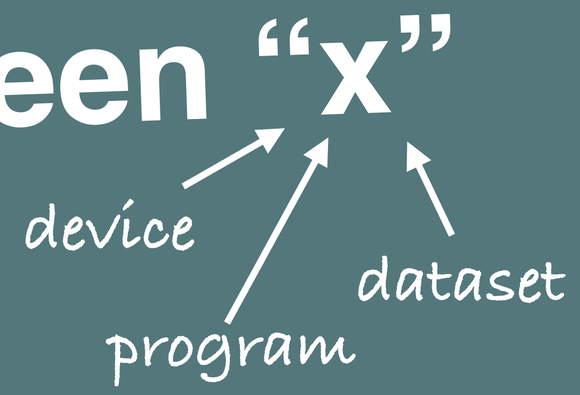
Pick the best one you tried

Collect data points
Extract “features”
Train machine learning classifier

Extract “features”
Input to classifier

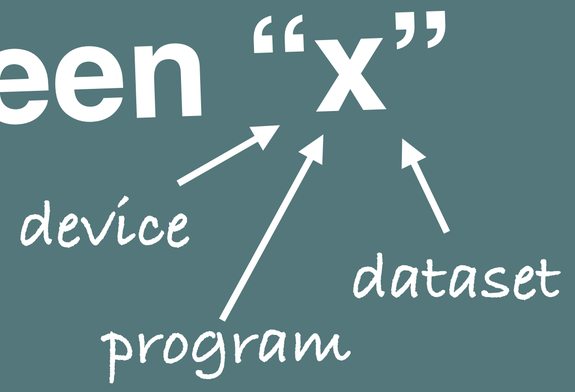
BETTER!

Can make *predictions* on unseen "x"



BETTER!

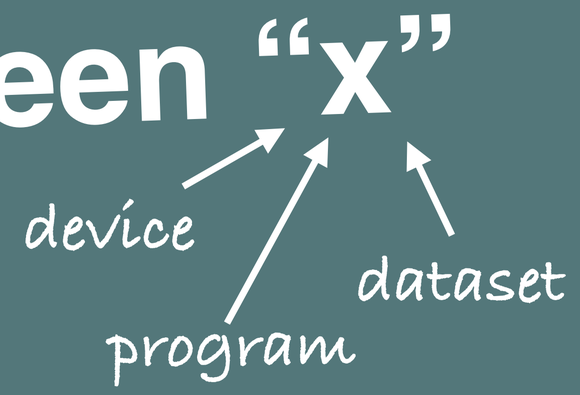
Can make *predictions* on unseen "x"



BETTER!

Still takes a loooong time

Can make *predictions* on unseen "x"



BETTER!

Still takes a loooong time

Requires a lot of code

Our wish list:

1. Reduce training costs
2. Reduce implementation costs
3. Minimise runtime overheads

A new approach ...

OmniTune

1. Allows *collaborative* performance tuning

Reduce training costs ✓

2. Provides re-usable *implementations*

Reduce implementation costs ✓

3. Provides lightweight runtime interface

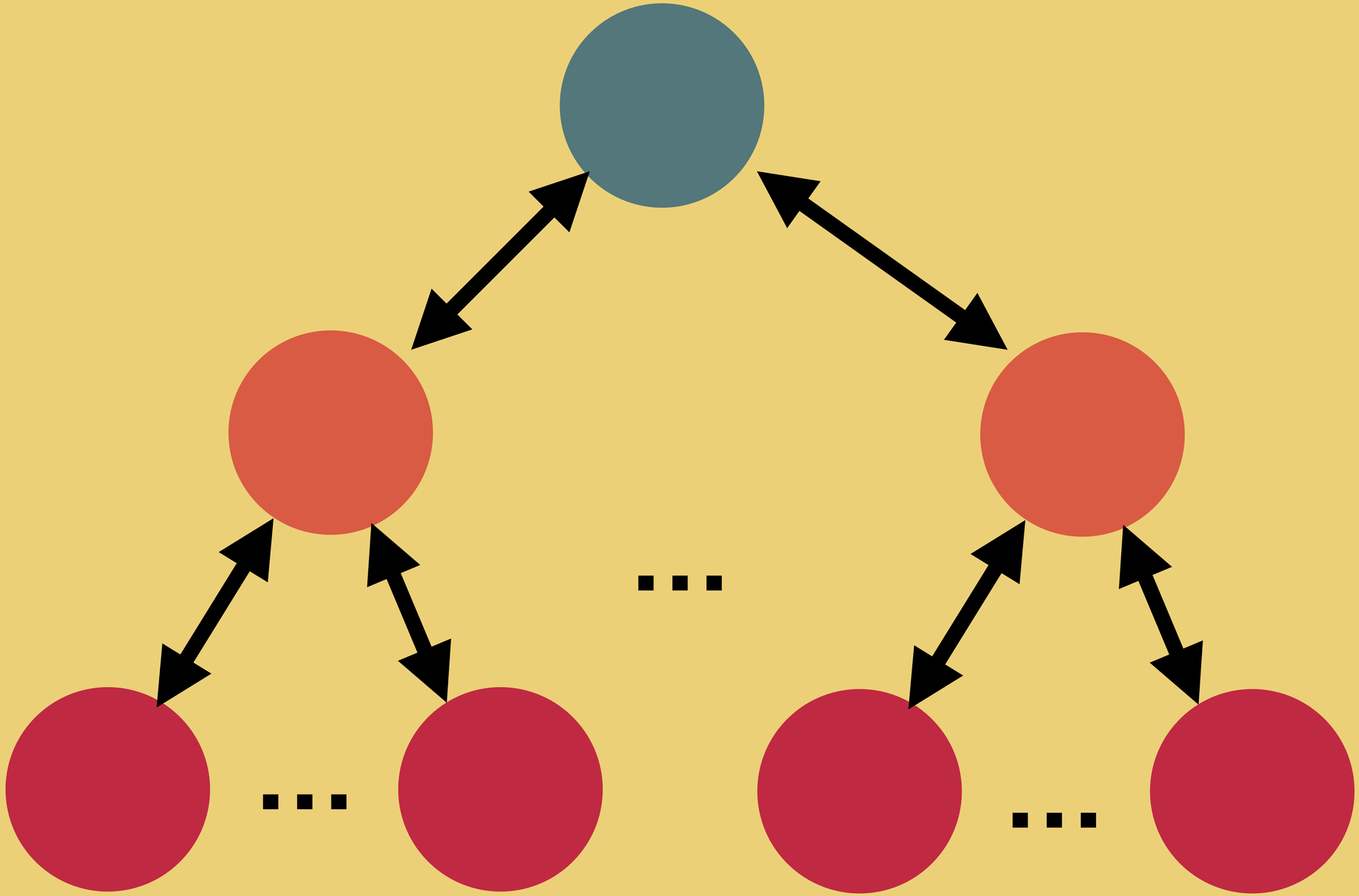
Minimise runtime overheads ✓

How does it work?

Remote

Servers

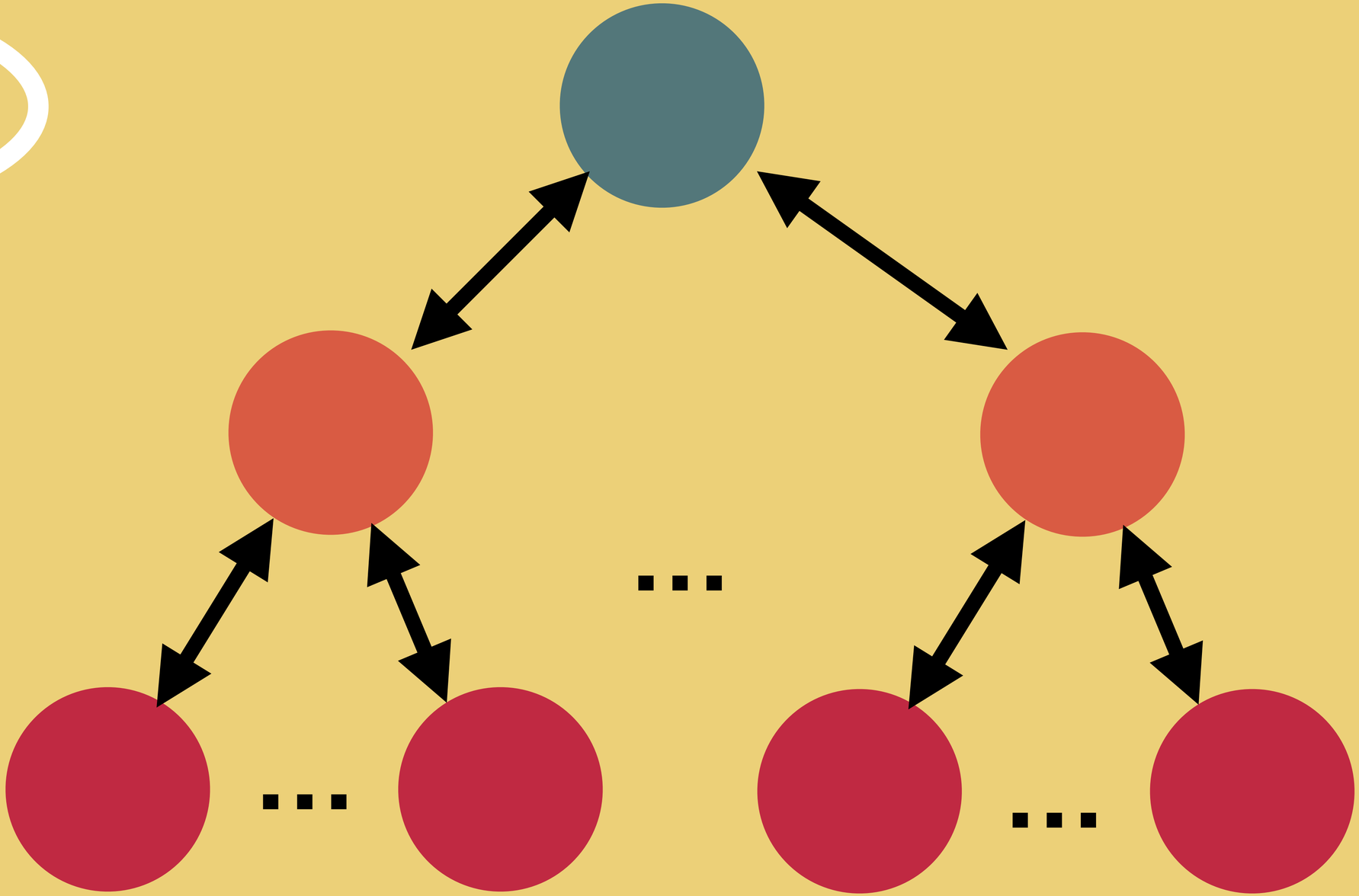
Clients



Remote

Servers

Clients



Remote

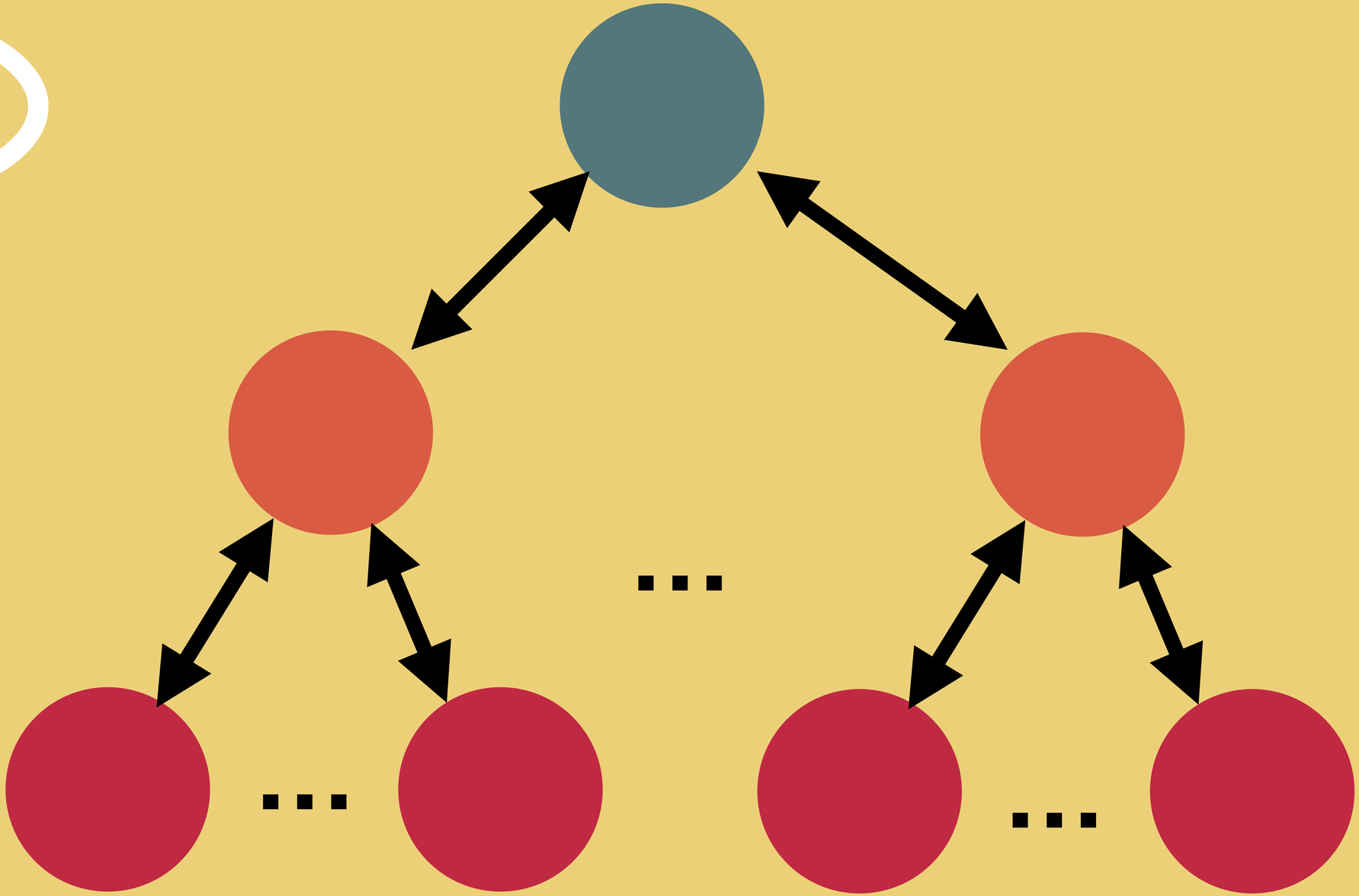
Book-keeper

Manages and stores training data

Remote

Servers

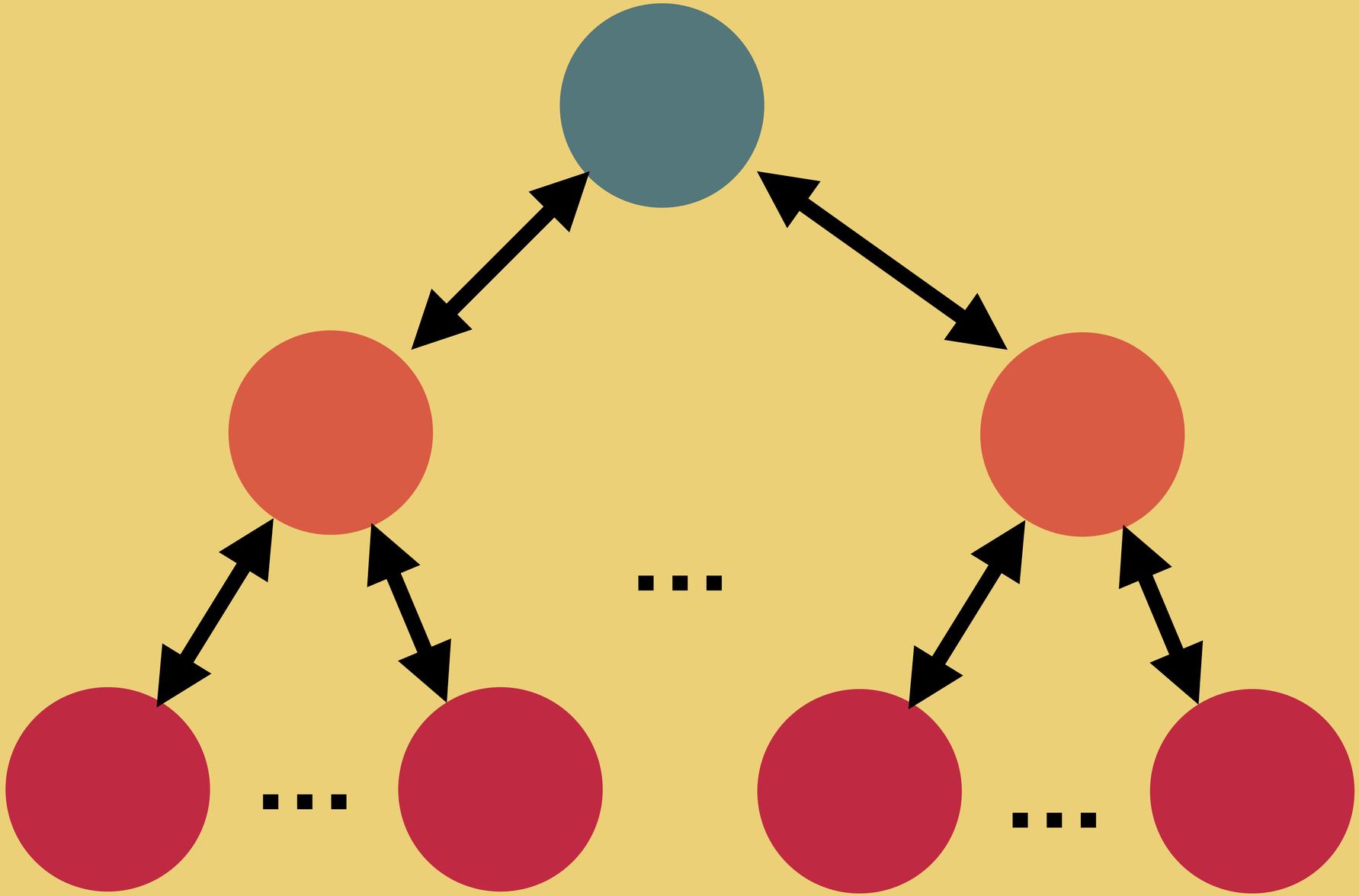
Clients

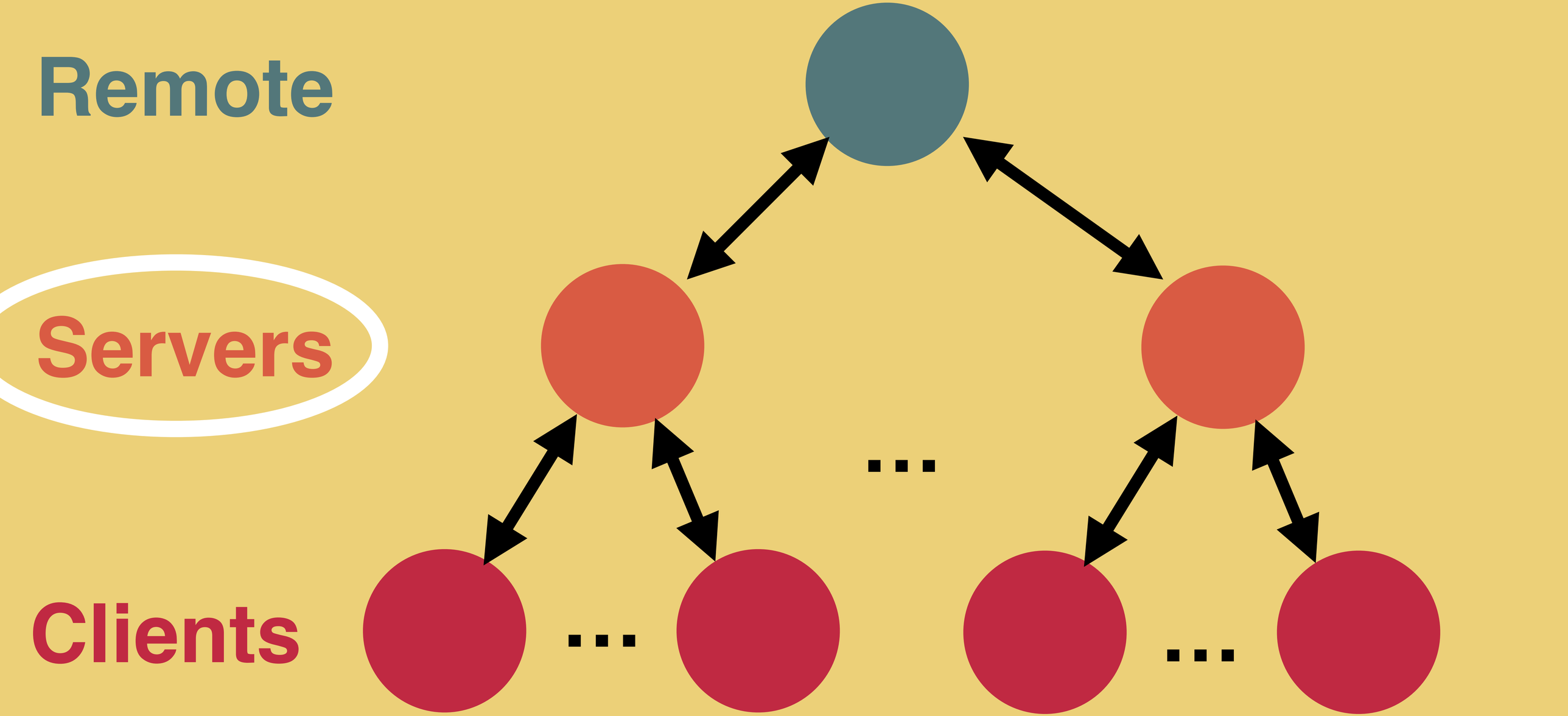


Remote

Servers

Clients

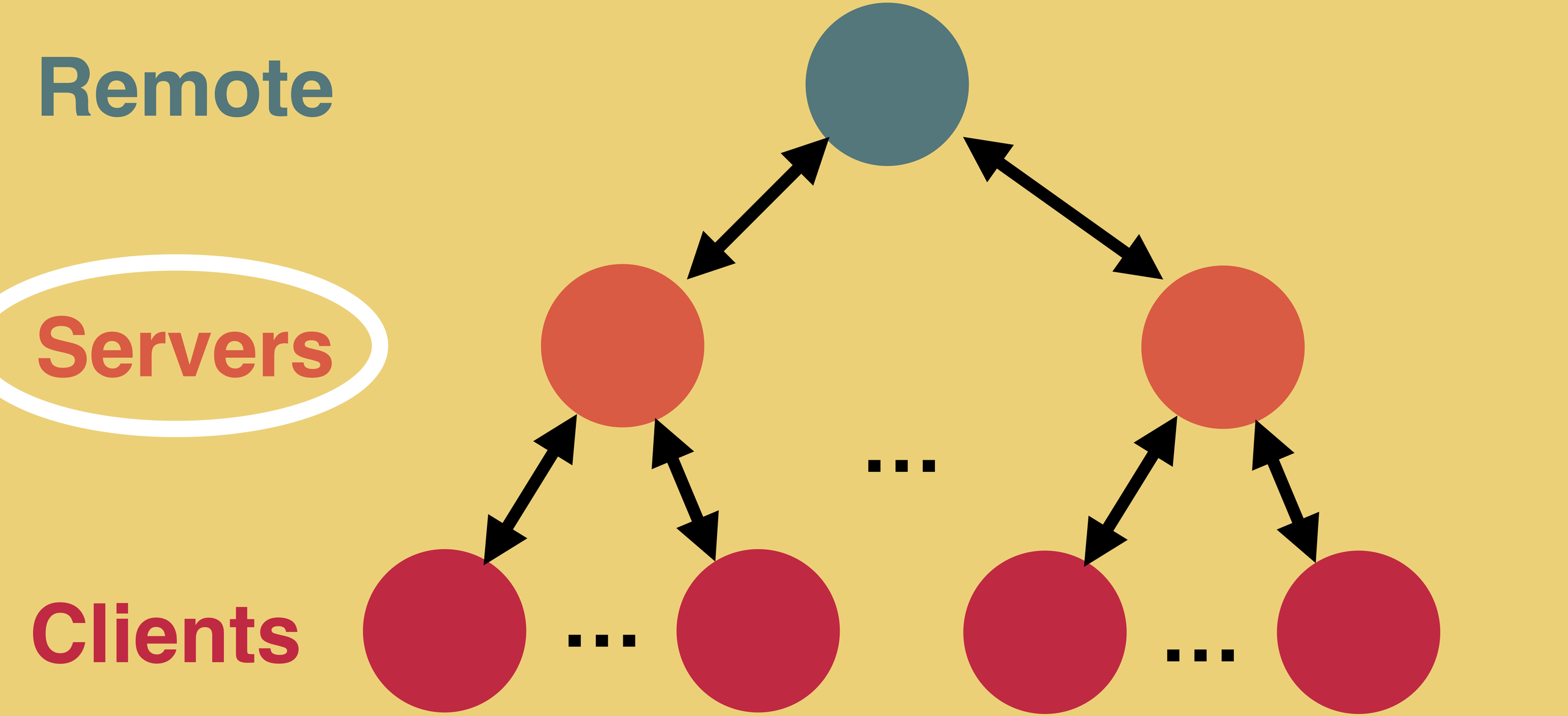



Remote
Servers
Clients
...
...
...

Servers

Autotuning engine

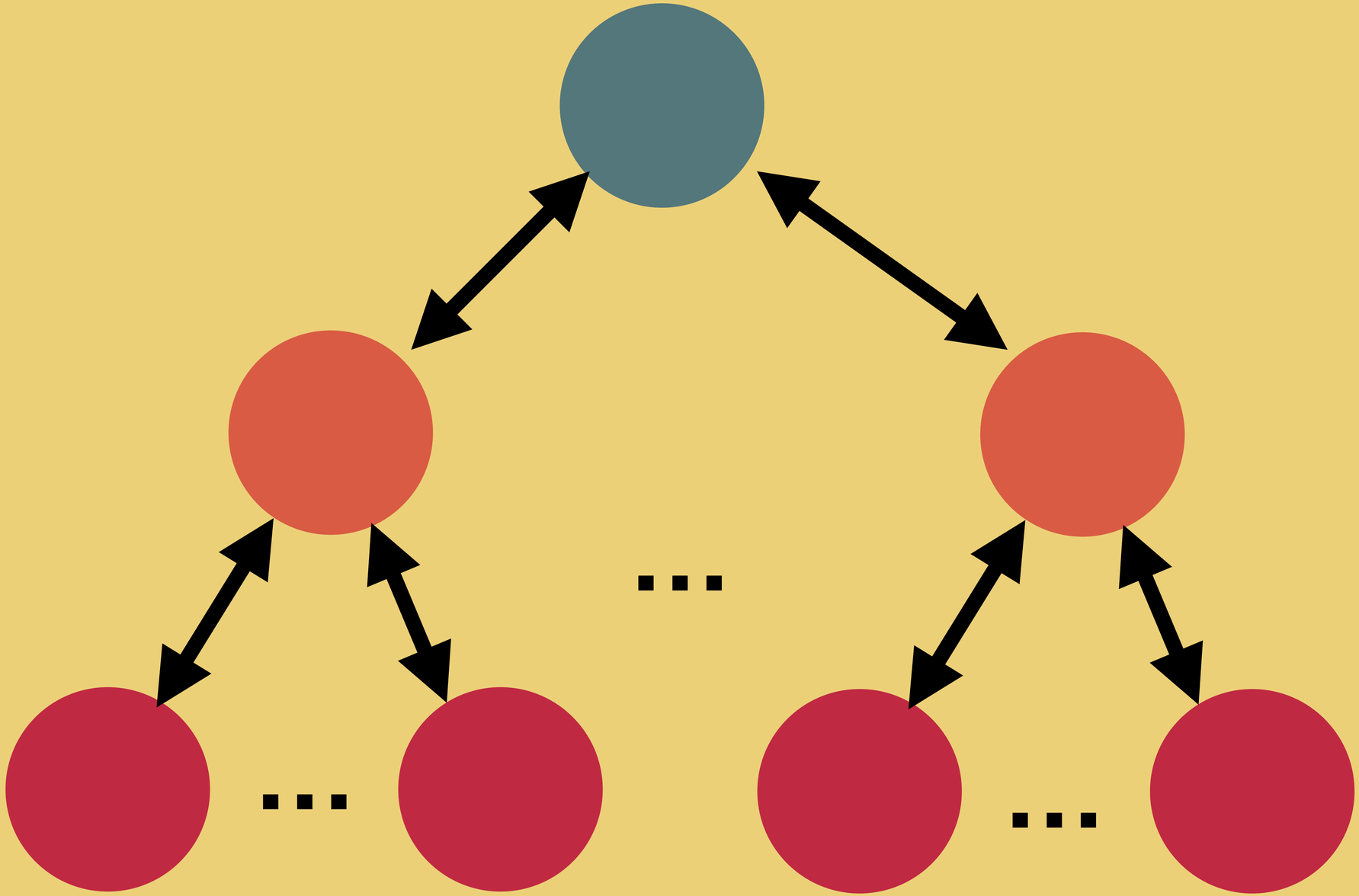
Performs machine learning

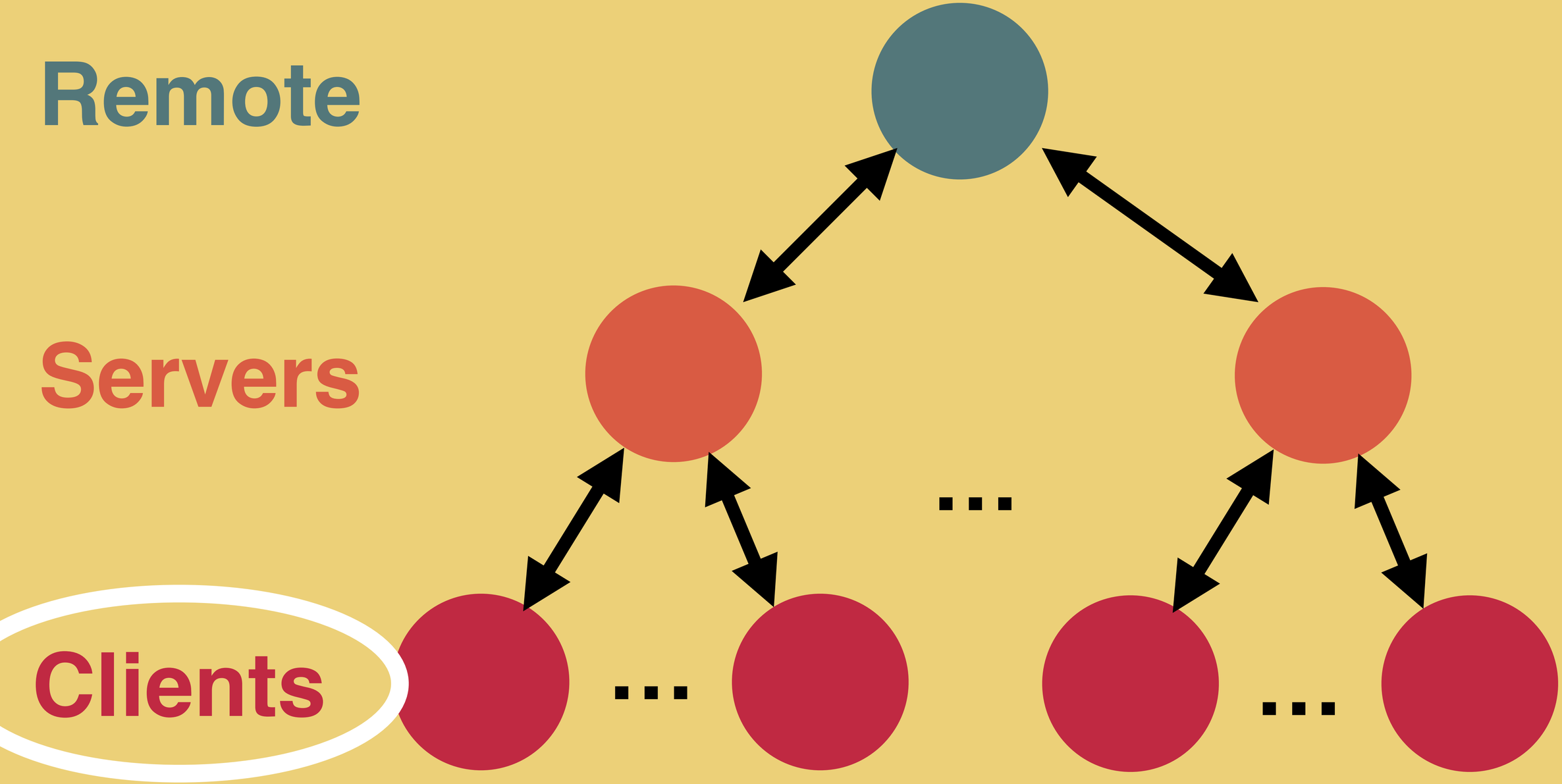

Remote
Servers
Clients
...
...
...

Remote

Servers

Clients

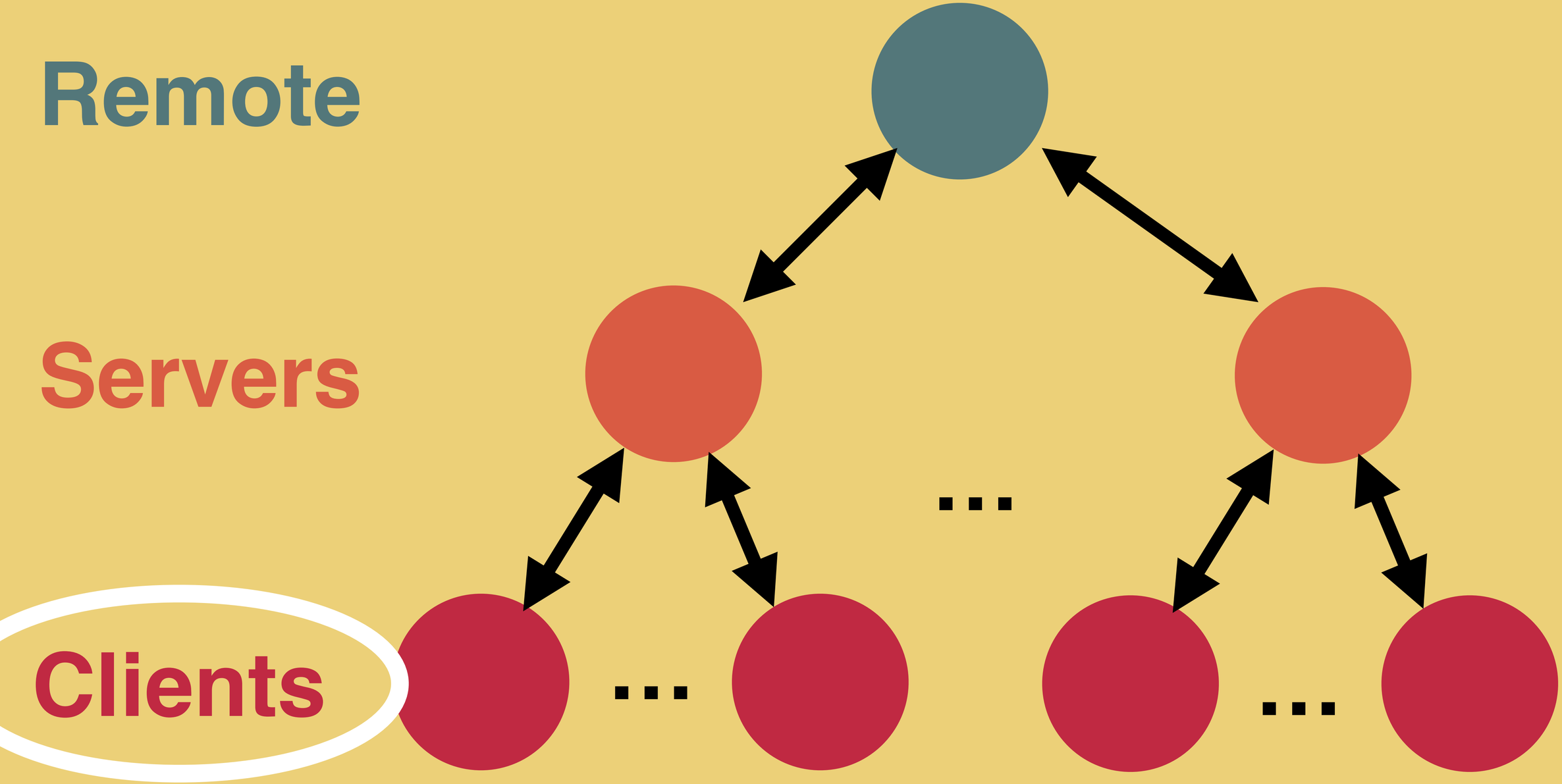



Remote
Servers
Clients
...
...
...

Target applications

Programs we want to tune

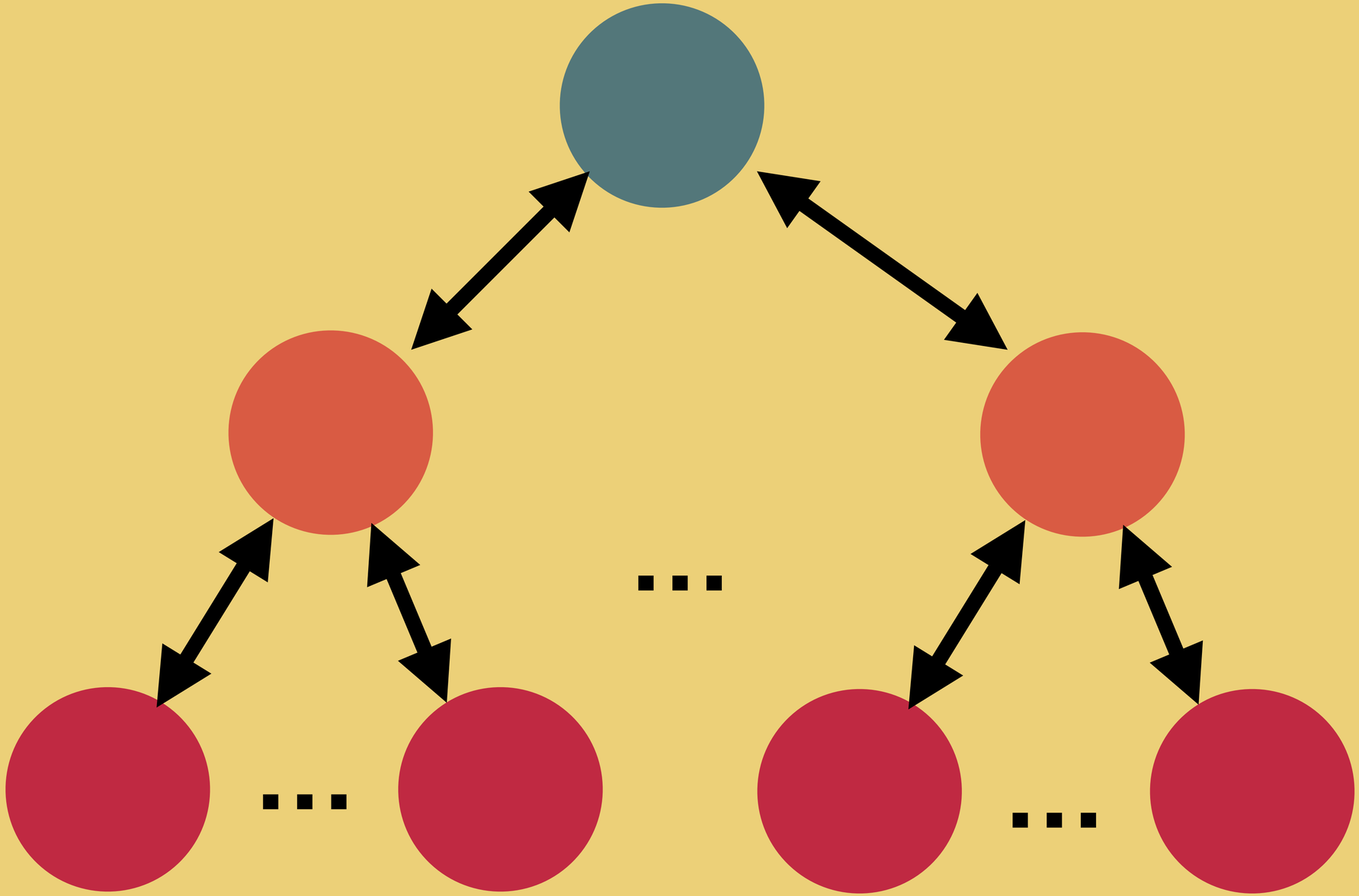
Clients

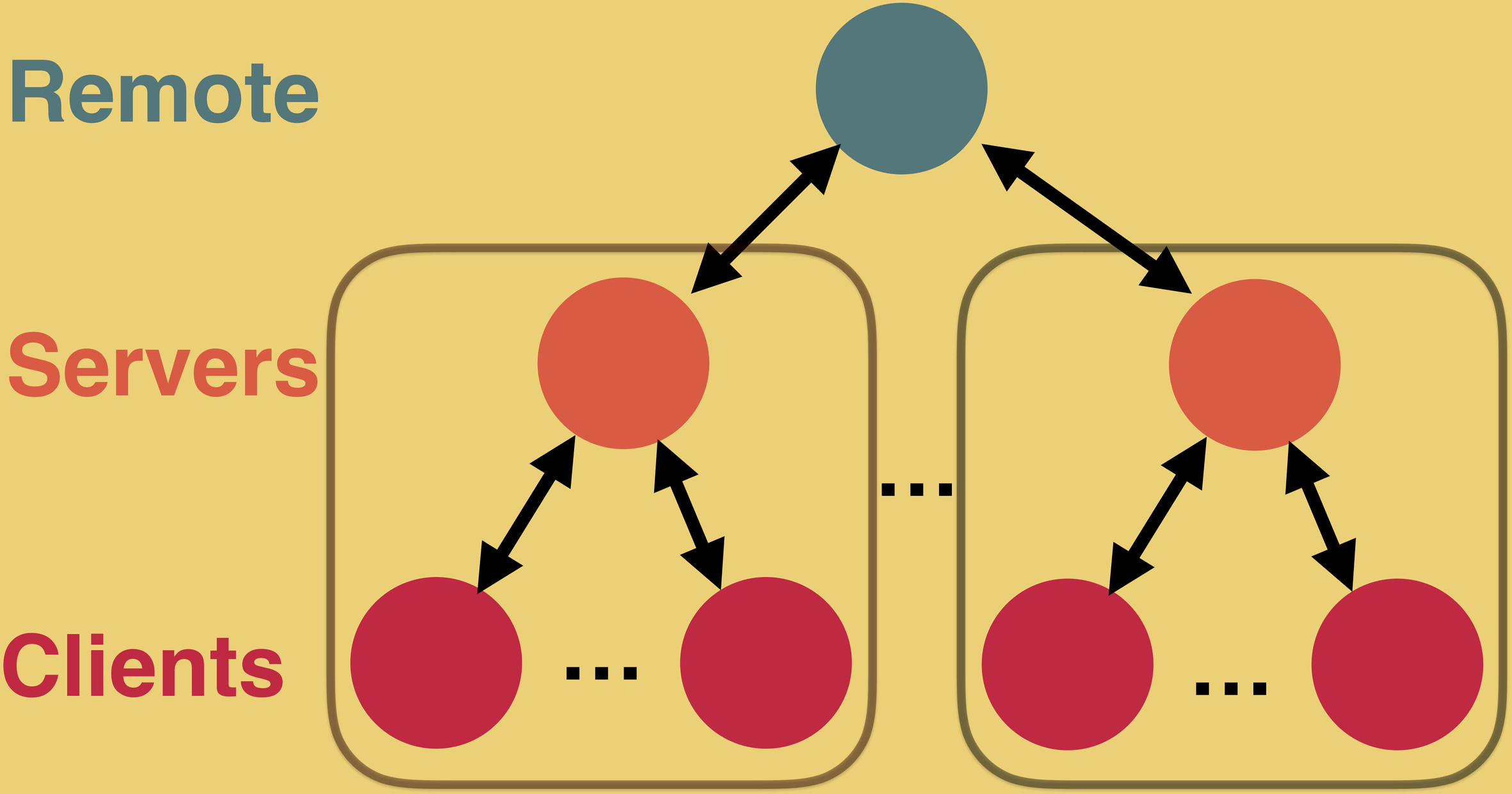

Remote
Servers
Clients
...
...
...

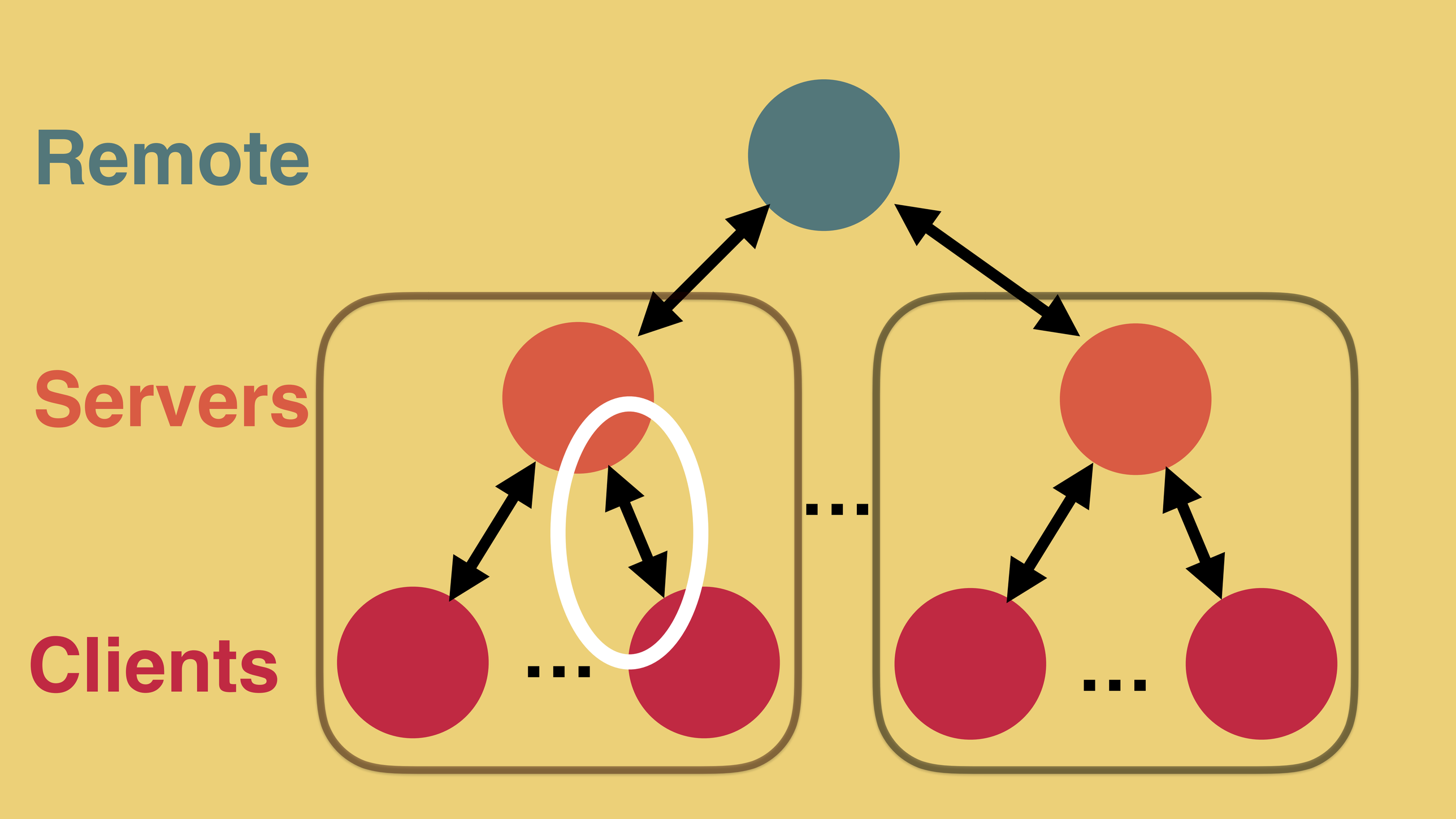
Remote

Servers

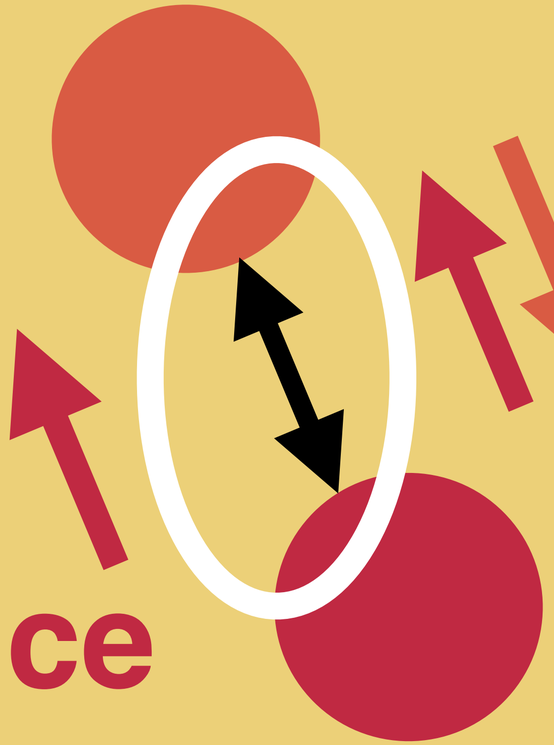
Clients




Remote
Servers
Clients
...
...
...

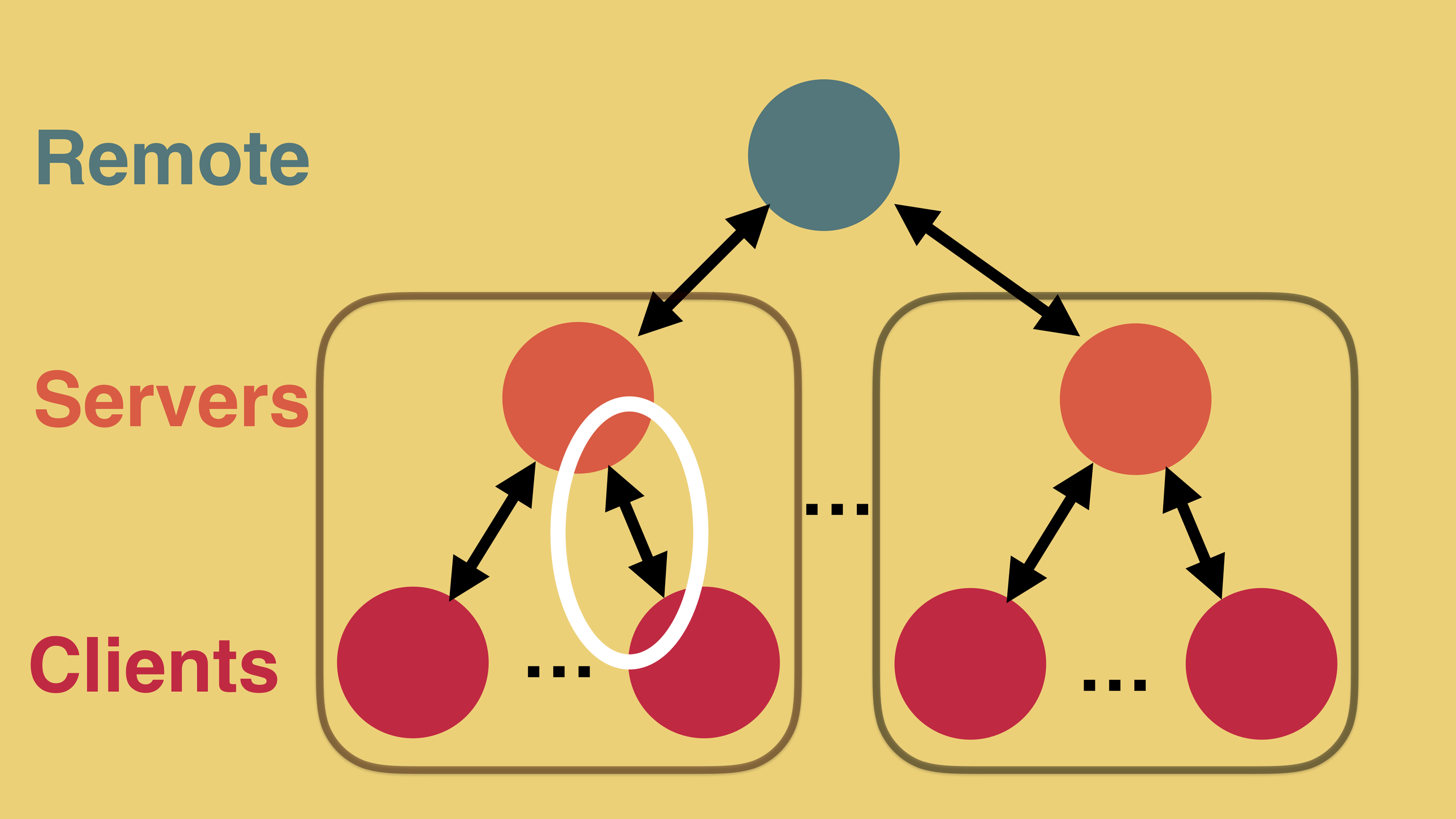

Remote
Servers
Clients
...
...
...

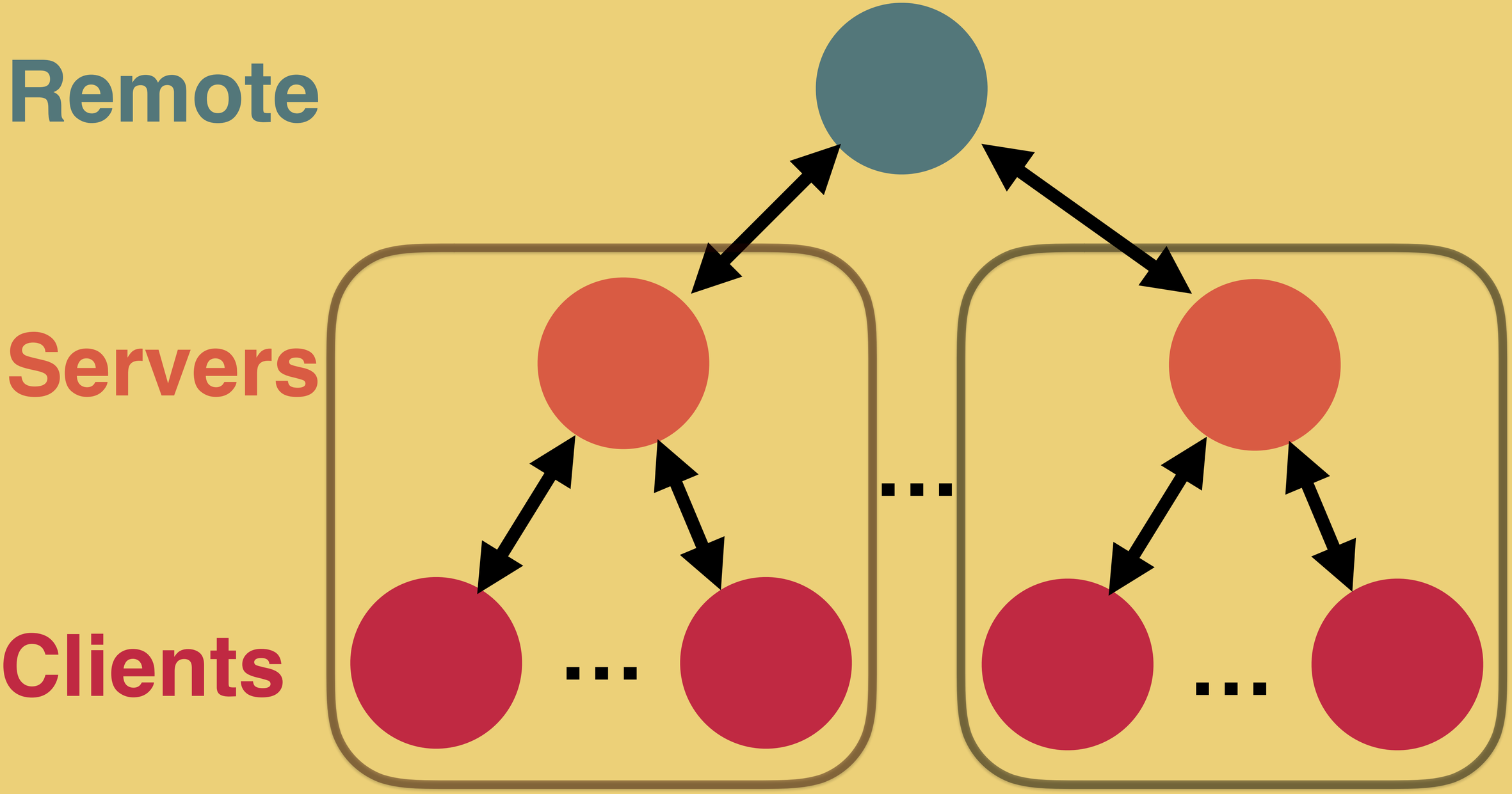
Features,
param,
performance

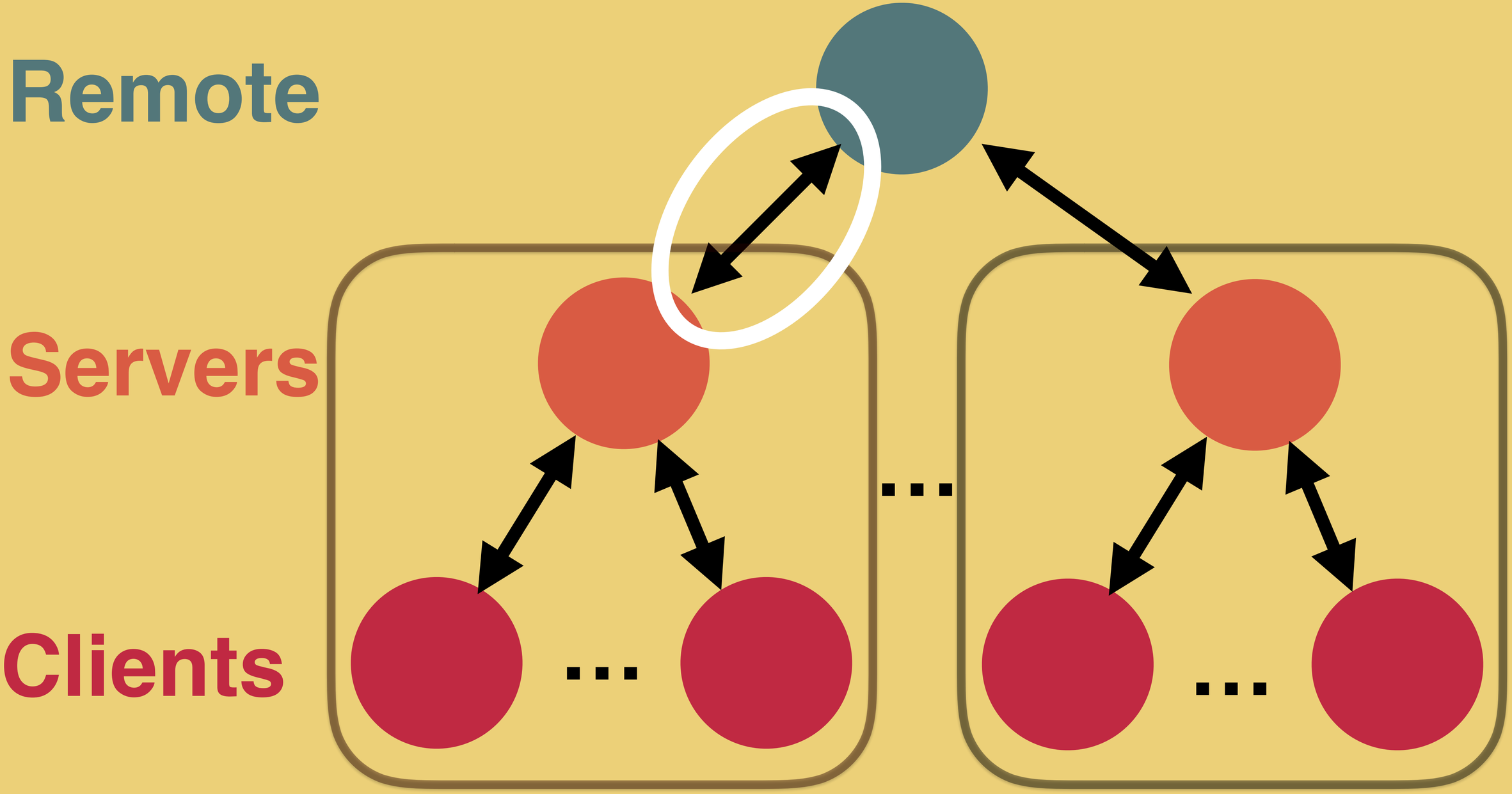


Features

Param

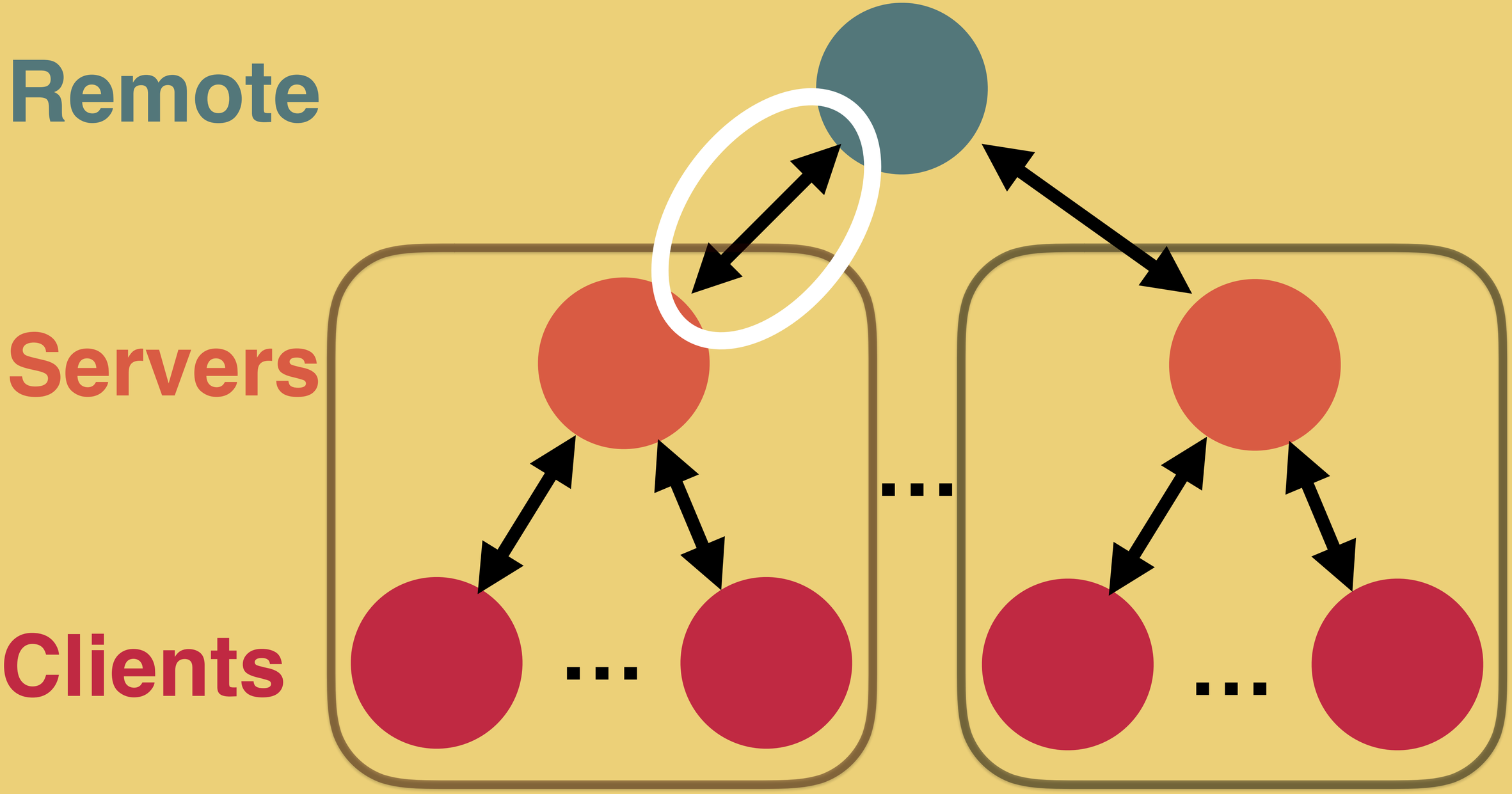

Remote
Servers
Clients
...
...
...

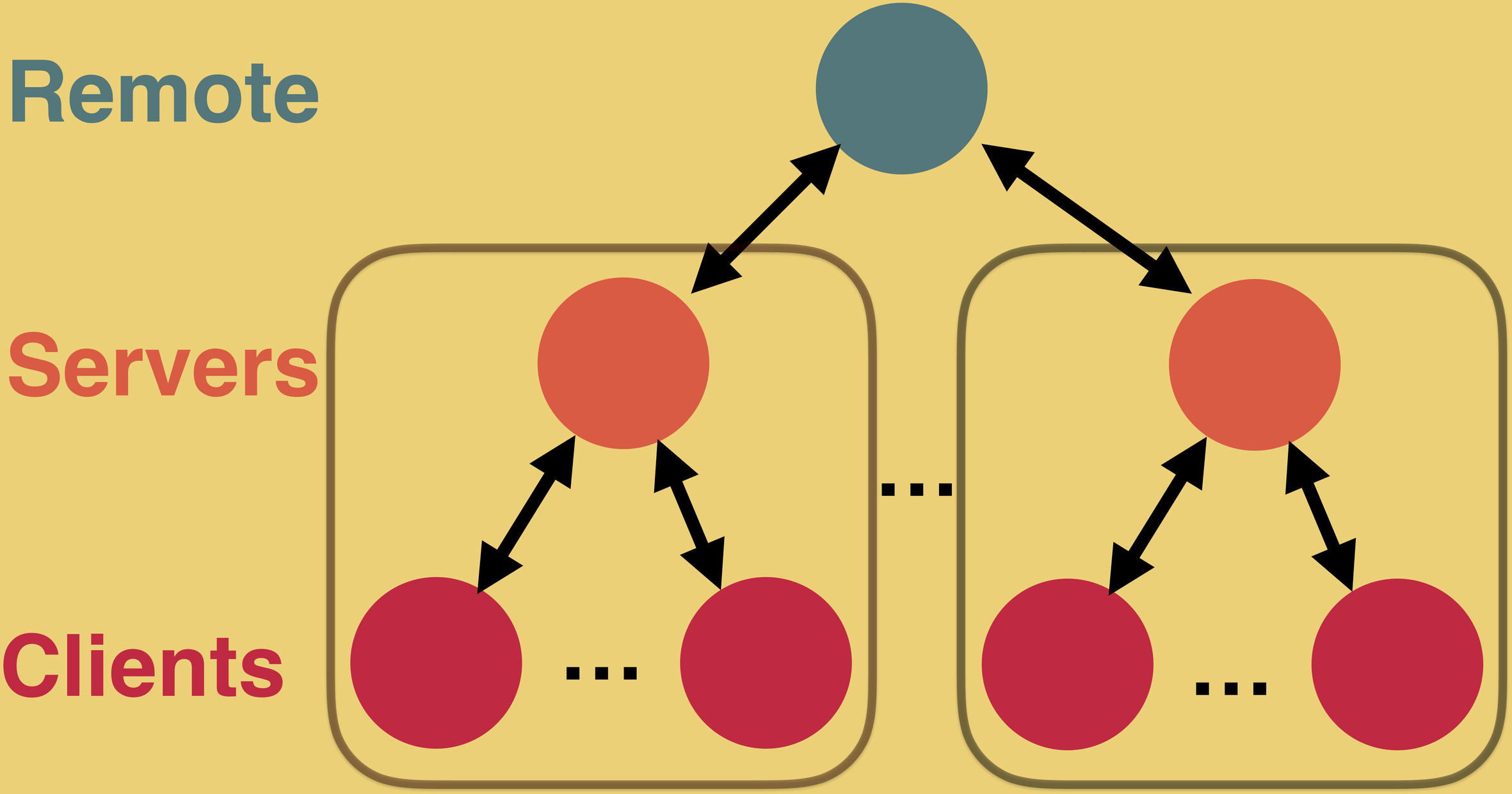

Remote
Servers
Clients
...
...
...


Remote
Servers
Clients

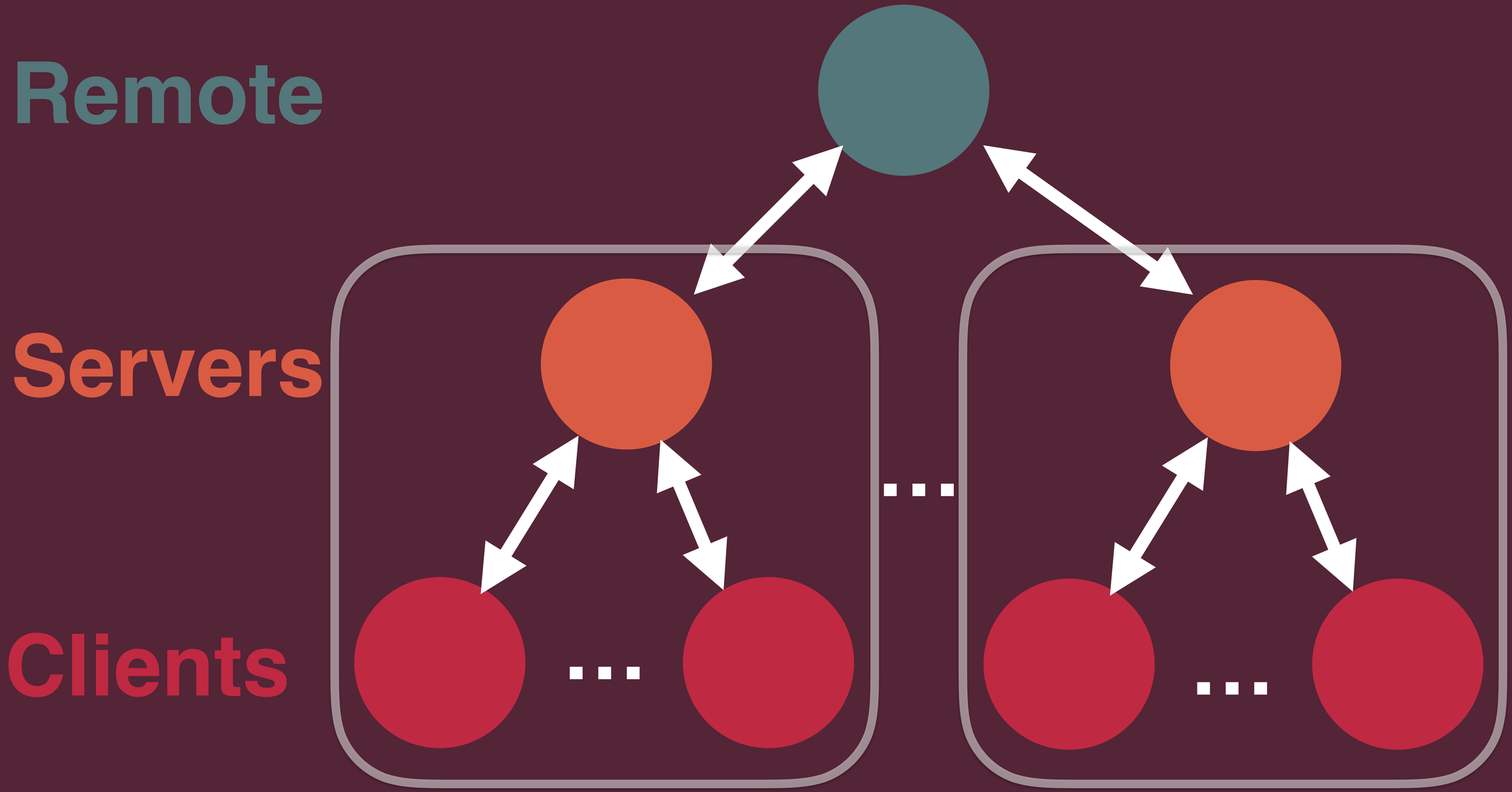
New training data

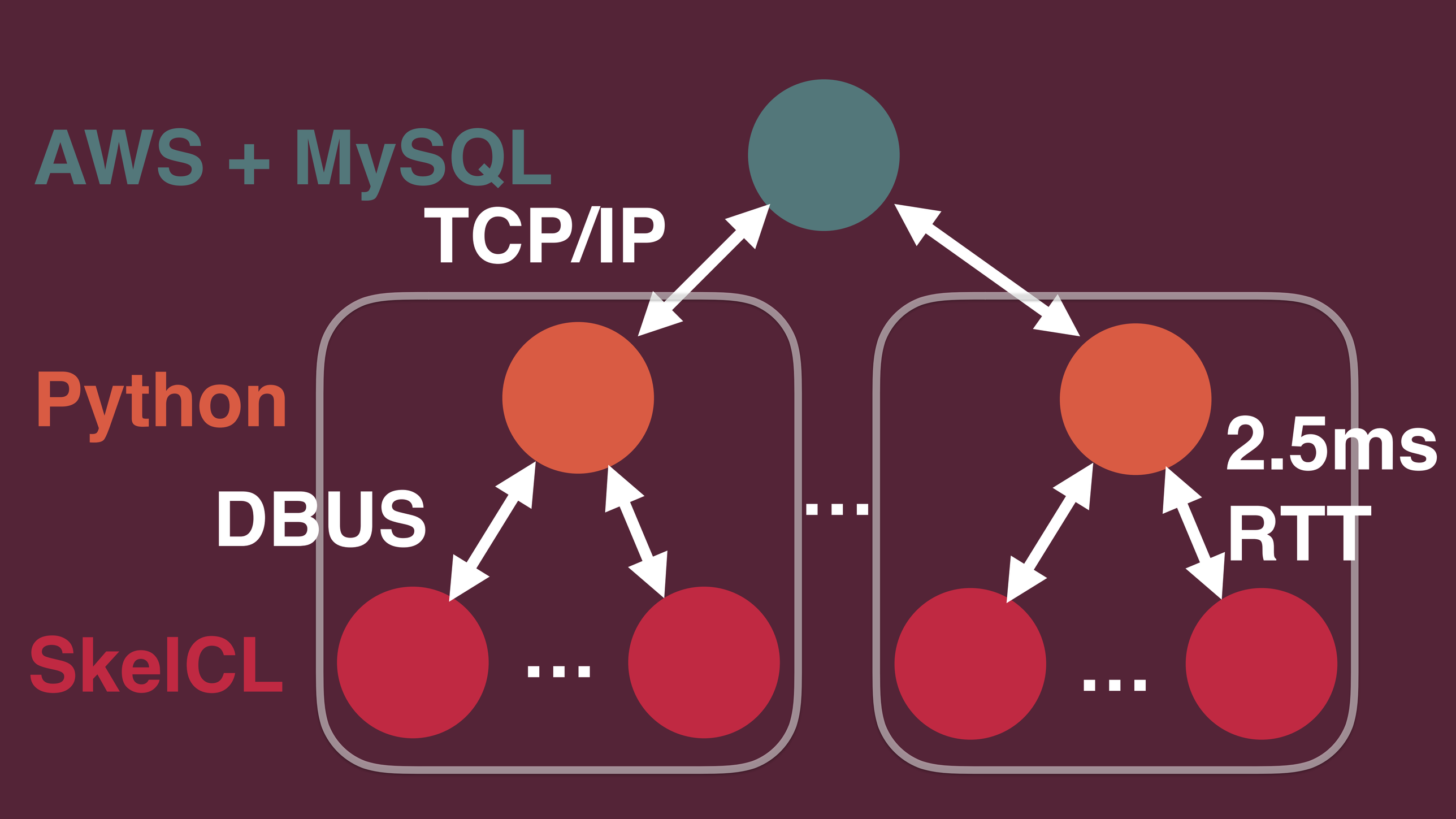
New training data


Remote
Servers
Clients


Remote
Servers
Clients
...
...
...

Implementation


Remote
Servers
Clients


AWS + MySQL
TCP/IP
Python
DBUS
SkelCL
2.5ms
RTT

Experiment

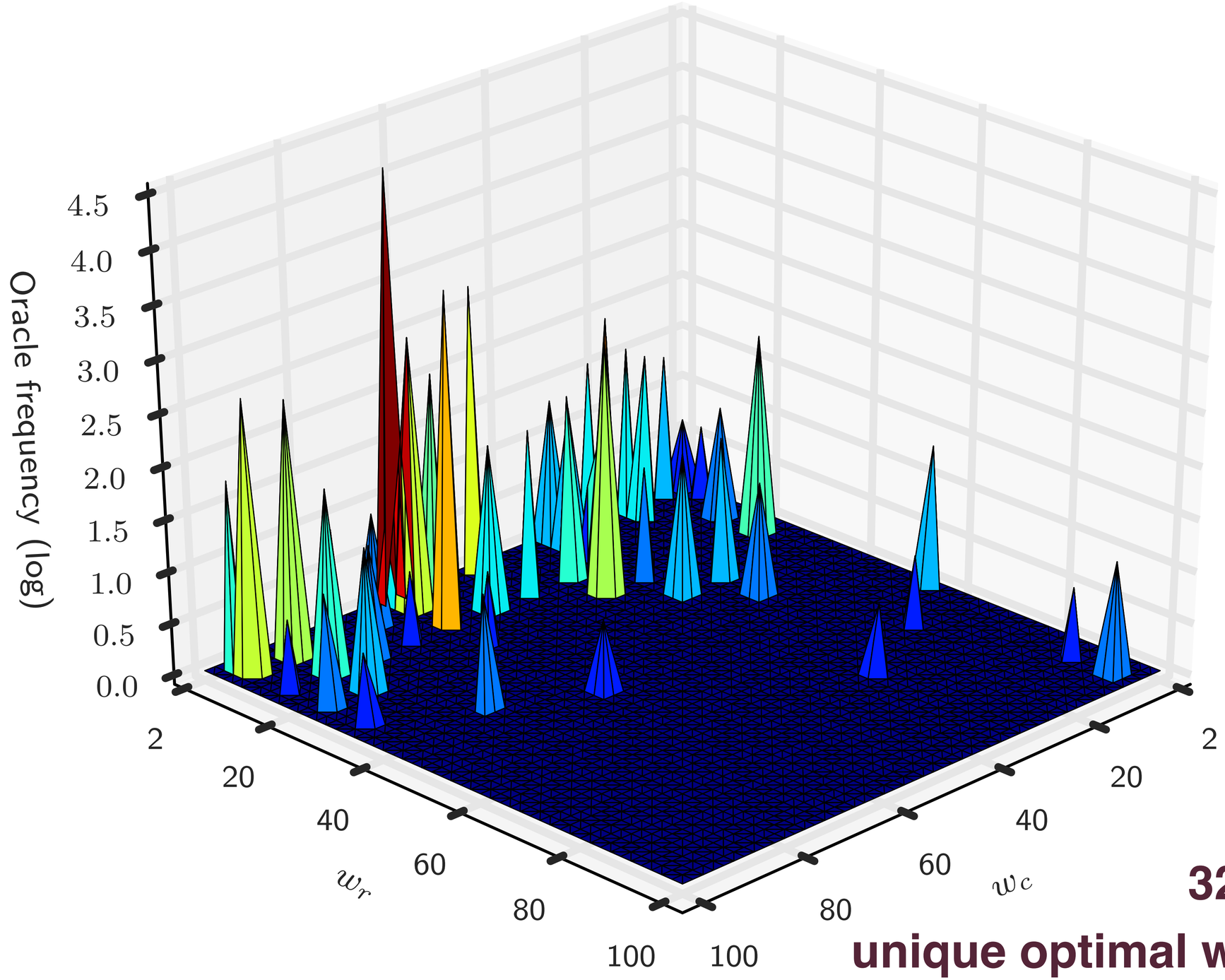
Setup:

6 stencil benchmarks + synthetic.

7 different GPUs & CPUs.

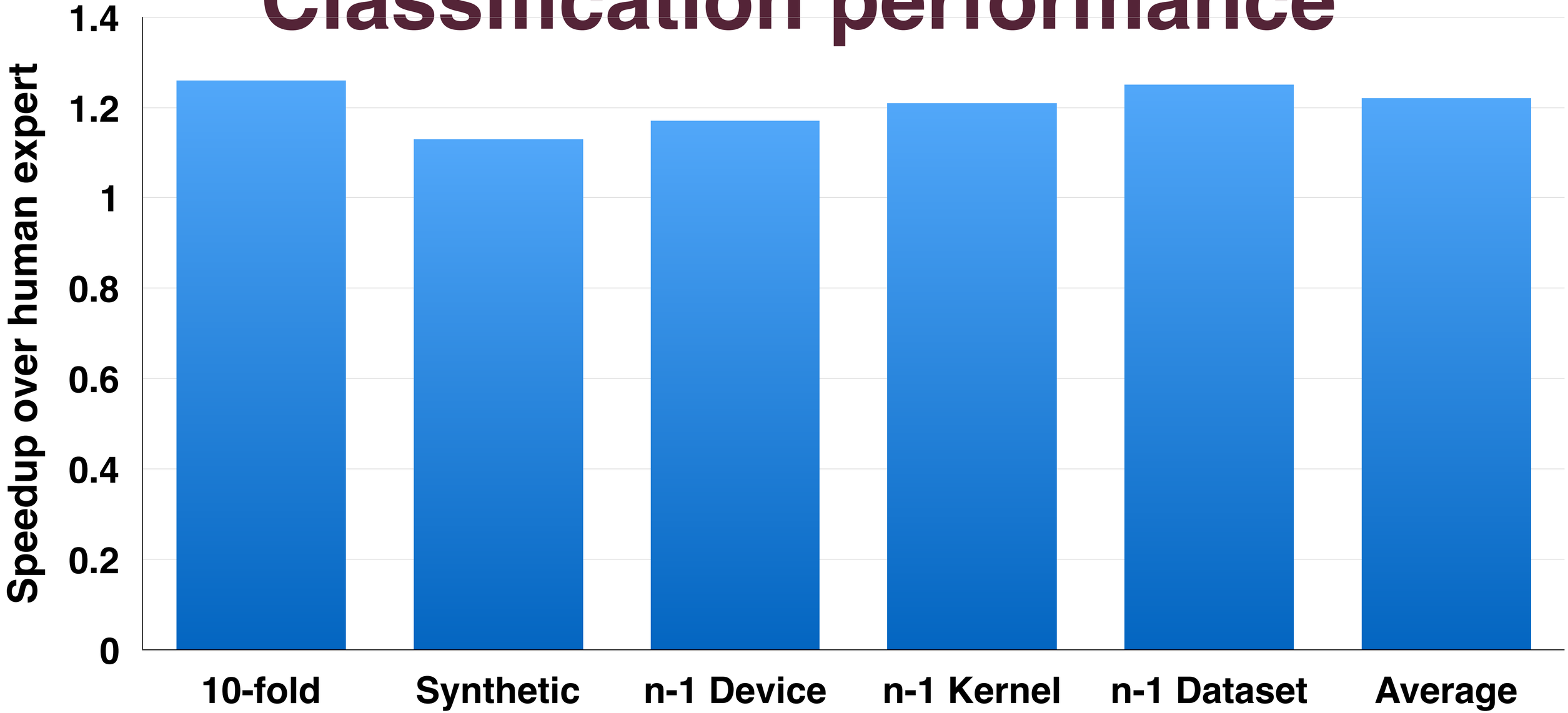
4 dataset sizes.

Exhaustive search of workgroup size space for each

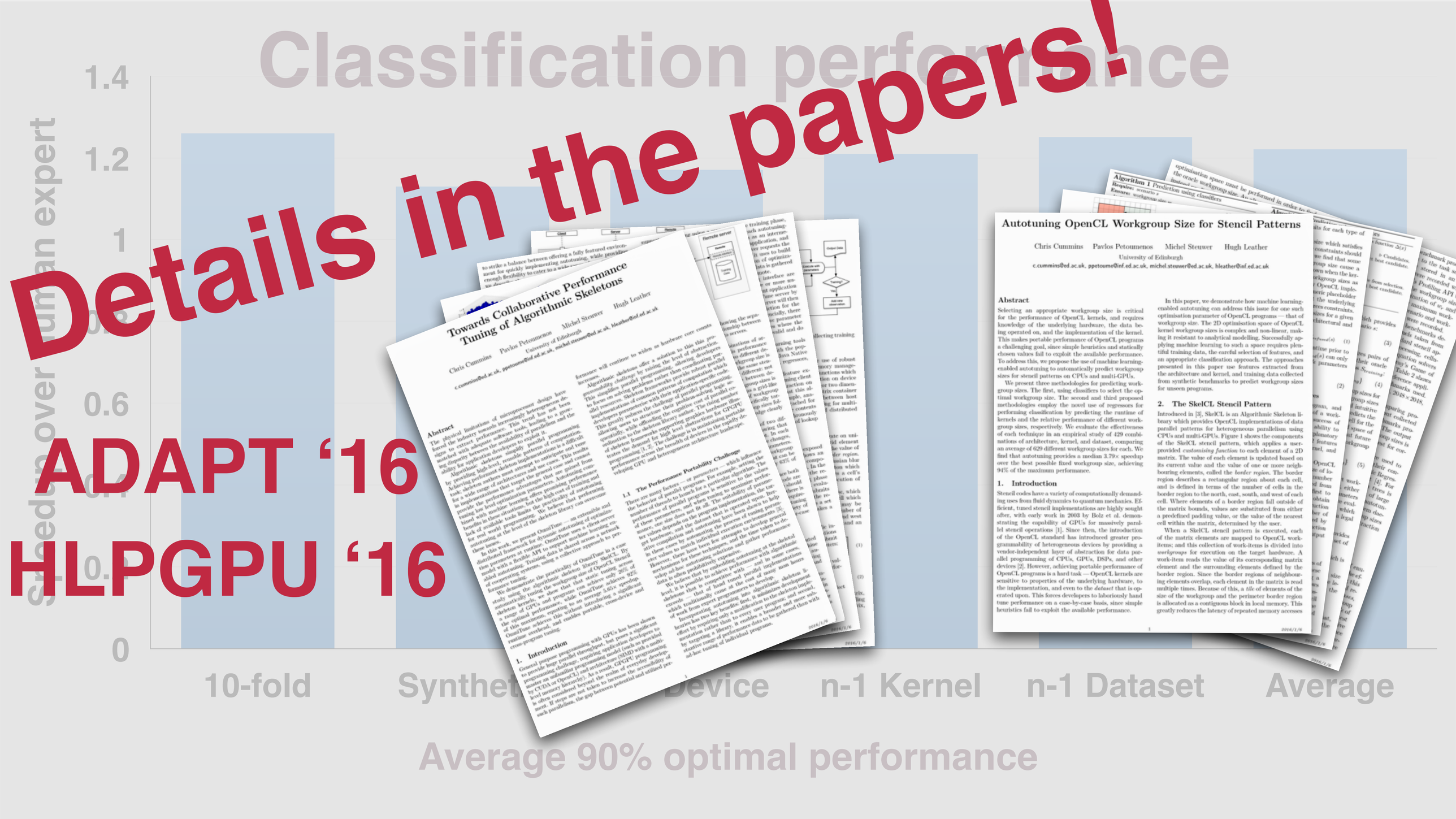


32% cases have unique optimal workgroup size

Classification performance



Average 90% optimal performance


Classification performance
Speedup over human expert
1.4
1.2
1
0.8
0.6
0.4
0.2
0
10-fold
Synthet
Device
n-1 Kernel
n-1 Dataset
Average
Average 90% optimal performance
Details in the papers!
ADAPT ‘16
HLPGPU ‘16
Towards Collaborative Performance Tuning of Algorithmic Skeletons
Chris Cummins Pavlos Petoumenos Michel Steuwer Hugh Leather
University of Edinburgh
Autotuning OpenCL Workgroup Size for Stencil Patterns
Chris Cummins Pavlos Petoumenos Michel Steuwer Hugh Leather
University of Edinburgh

Conclusions

High level code must compete with low level on *performance*

That means *automating* the kind of tuning which is typical of low level

I designed a framework for doing this using machine learning

Demonstrated using SkelCL stencils

Achieves average 1.22x speedup over *human expert*

Autotuning and *Algorithmic Skeletons*