NodeJS Supercomputing

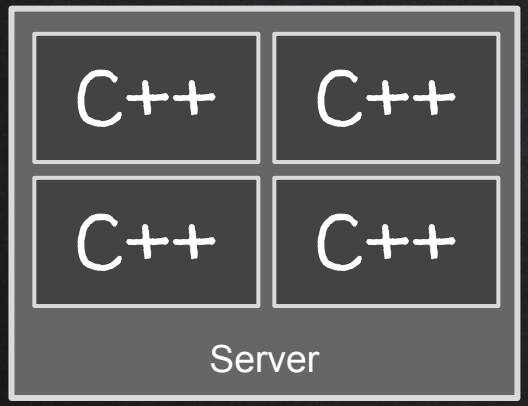
Oliver Rumbelow

http://arstechnica.com/business/2011/09/30000-core-cluster-built-on-amazon-ec2-cloud/

$1,279-per-hour, 30,000-core cluster built on Amazon EC2 cloud

EMBARRASSINGLY PARALLEL

- 3,809 compute instances
- each with eight cores
- ran across data centers in three Amazon regions
- ran for about seven hours



OpenMP

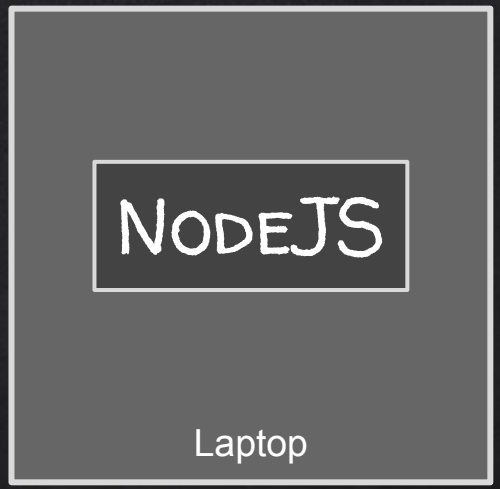
The OpenMP® API specification for parallel programming

...multi-platform shared-memory parallel programming in C/C++...

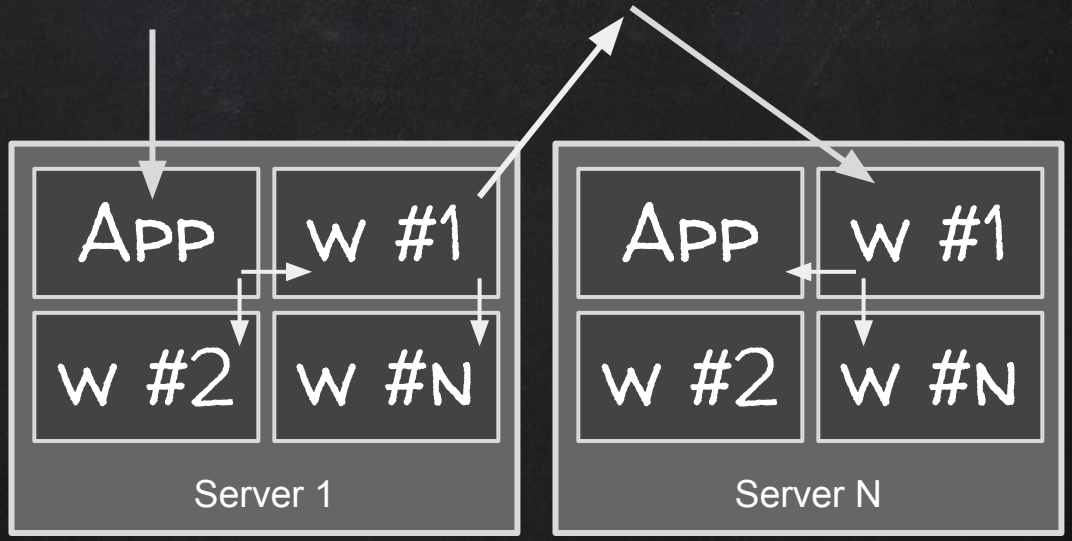
Bespoke

High Barrier To Entry

Expensive


NODEJS
Laptop

```
var multi = require('./multi.js');
if (multi._isMaster) return;
multi.loadBalancer('localhost');
```


App
w #1
w #2
w #N
Server 1
App
w #1
w #2
w #N
Server N

Multithreaded NodeJS

Easy!

```
require("cluster");
```

```
var cluster = require('cluster');

if (cluster.isMaster) {
  cluster.fork();
}

console.log("Ready"); // Occurs twice
```


0
Master
Worker #1

```
var cluster = require('cluster');

if (cluster.isMaster) {
  cluster.setupMaster({ exec: module.filename });
  var numCPUs = require('os').cpus().length;
  while (numCPUs--) {
    cluster.fork();
  }
  return;
}
```


Worker #1
Worker #2
Worker #3
Worker #N

MOVING DATA BETWEEN PROCESSES


1
Worker #1
Worker #2

1

Easy?

```
require("cluster");
```


```
if (cluster.isMaster) {
  var worker = cluster.fork();
  worker.send('hi there');

} else if (cluster.isWorker) {
  process.on('message', function(msg) {
    process.send(msg);
  });
}
```


1
1MB
x100
Worker #1
Worker #2

WORKER.SEND();

12,303 MILLISECONDS

WORKER.SEND();

16.25 MB/s

1

REDIS.SET() + REDIS.GET()

REDIS.PUBLISH() + REDIS.SUBSCRIBE()

5,388 MILLISECONDS

1

FS.READ() + FS.WRITE()

1173 MILLISECONDS

TCP

774 MILLISECONDS

FS.READ() + FS.WRITE()

/TMP (TMPFS RAM DISK)

395 MILLISECONDS

UNIX SOCKET

242 MILLISECONDS

1

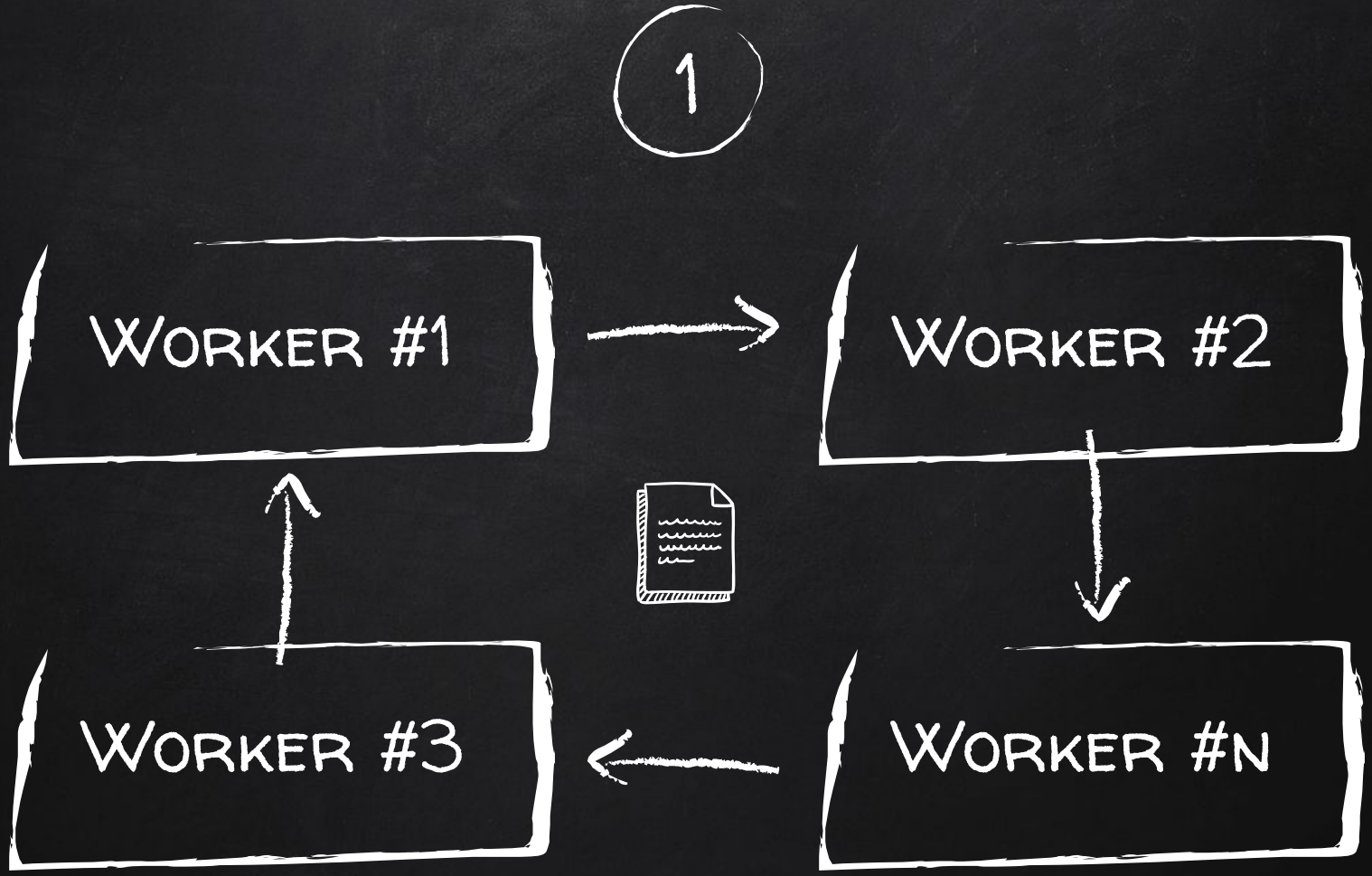
UNIX SOCKET

826 MB/s

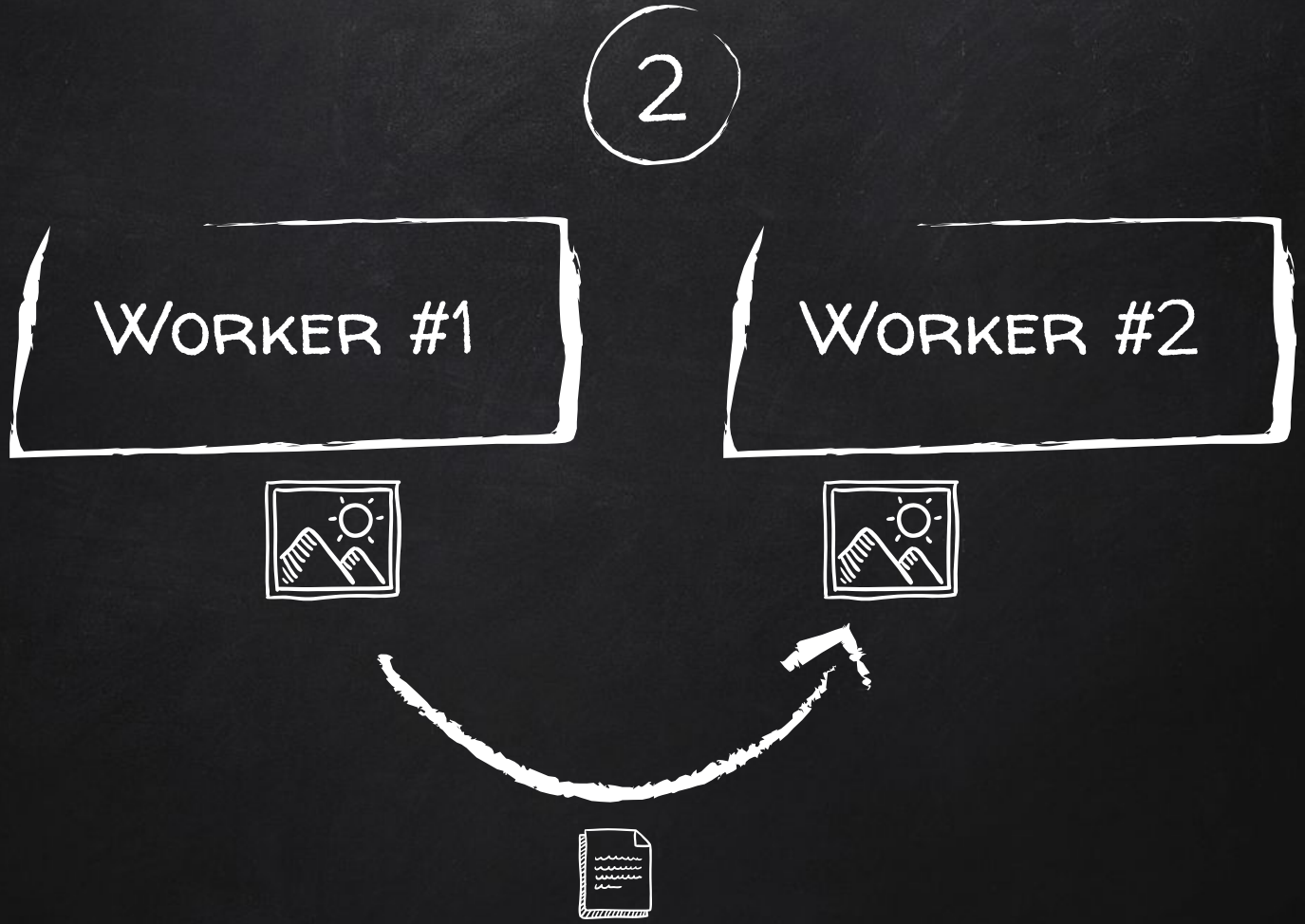
!! UNIX SOCKET !!

! TCP !

https://github.com/RIAEvangelist/node-ipc


1
Worker #1
Worker #2
Worker #3
Worker #N

SERIALISING COMPLEX OBJECTS


2
Worker #1
Worker #2

SERIALIZE NODE-SERIALIZE SERIALIZE-OBJ SERIALIZE-ERROR EXPRESS-SERIALIZE FORM-SERIALIZE BACKBONE-SERIALIZE SERIALIZE-OBJECT VDOM-SERIALIZE SERIALIZE-JAVASCRIPT SERIALIZE-TO-JS K-SERIALIZE-OBJECT SERIALIZE-LIKE-PHP SERIALIZE-FOR-XHR COMMONFORM-SERIALIZE EVAL-SERIALIZE SERIALIZE-DEPTREE SERIALIZE-SELECTION SERIALIZE-FLUX SERIALIZE-ELEM SERIALISE-JS SERIALISE-PROMISES SERIALISED-ERROR OBJECT-TOJSON SYSLOG-SERIALIZE SERIALIZE-DOM DOM-SERIALIZE PHP-SERIALIZE SERIALIZE-JS JSON-SERIALIZE SERIALIZE-SVG-PATH MGSCARP-OPENINGHOURS-SERIALIZE EVAL-SERIALIZE-TYPED-ARRAY EVAL-SERIALIZE-BUFFER USEFUL-WIND-SERIALIZE EVAL-SERIALIZE-DATE EVAL-SERIALIZE-POSITIVE-INFINITY EVAL-SERIALIZE-NEGATIVE-INFINITY SERIALIZE-FORM SERIALIZE-STL SERIALIZE-ARRAY .....


```
var circular = { };
circular.loop = circular;
console.log(circular);

> { loop: [Circular] }
```


```
var circular = { };
circular.loop = circular;
console.log(JSON.stringify(circular));

> TypeError: Converting circular structure to JSON
>    at Object.stringify (native)
```


```
var a = { foo: "bar" };
var b = { first: a, second: a };

console.log(b);

> { first: { foo: 'bar' }, second: { foo: 'bar' } }
```


```
var moment = require("moment");

var now = moment();
console.log(JSON.stringify(now));

> "2016-01-06T20:25:13.140Z"
```


```
var myClass = function() {
  this.foo = "bar";
};
var myInstance = new myClass();

console.log(myInstance);

> { foo: 'bar' }
```

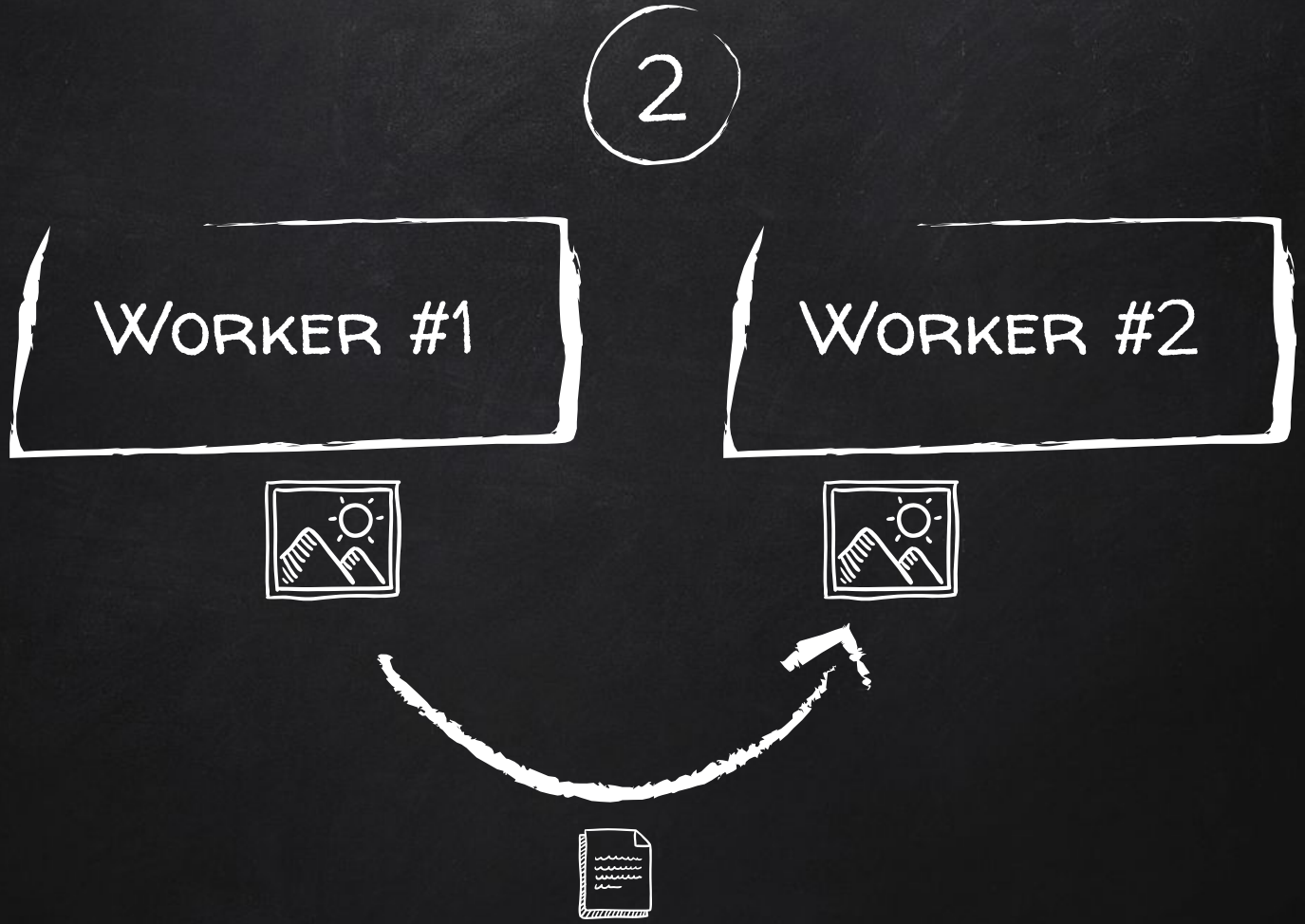
2

CLOSURES

2

```
var getDataBuffer = function() {
  var last = null;
  return function(param) {
    var returnValue = last;
    last = param;
    return returnValue;
  };
};
```

```
var buffer = getDataBuffer();
console.log(buffer(0)); // null
console.log(buffer(1)); // 0
console.log(buffer(2)); // 1
```

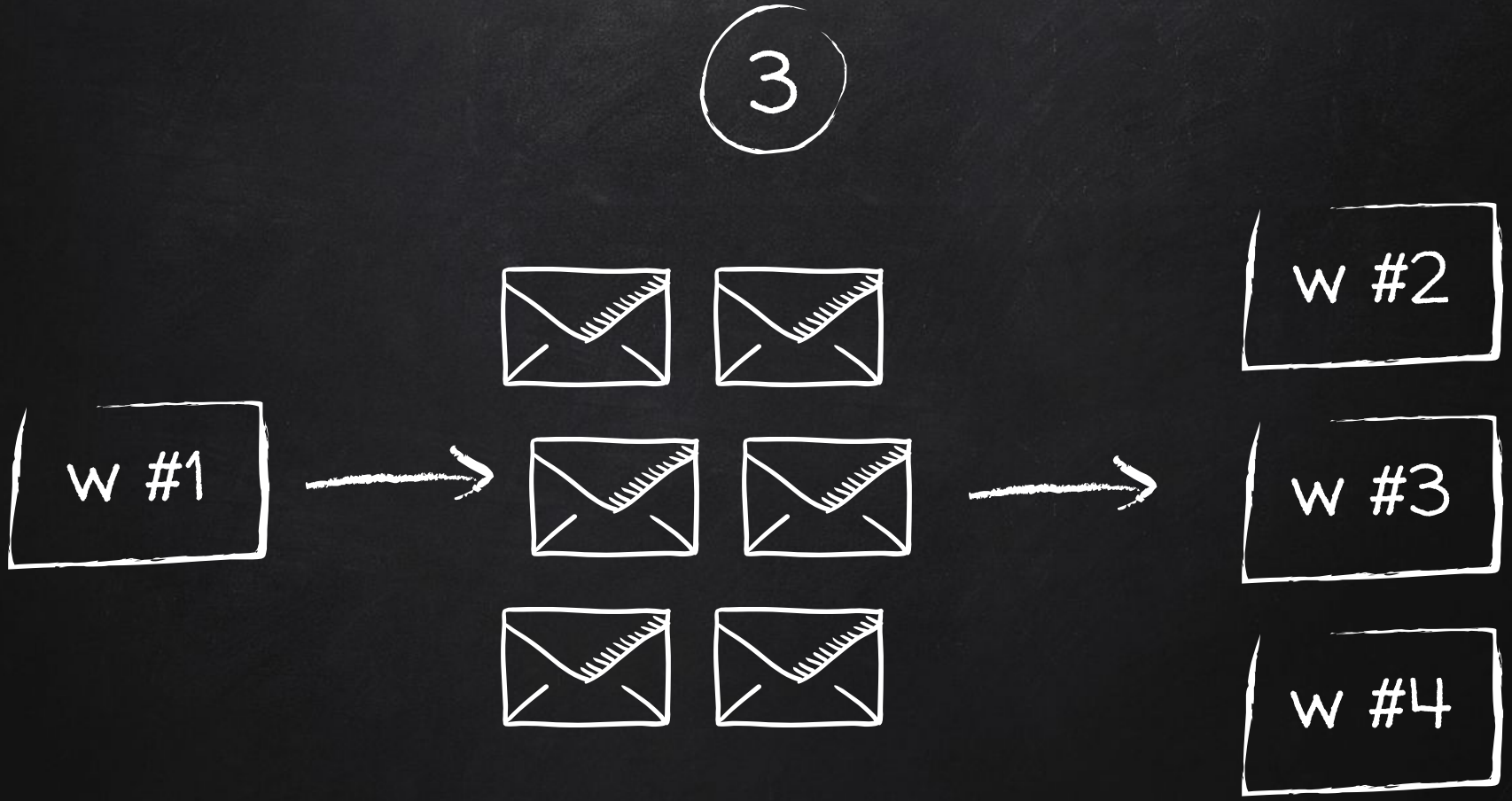

2
Worker #1
Worker #2

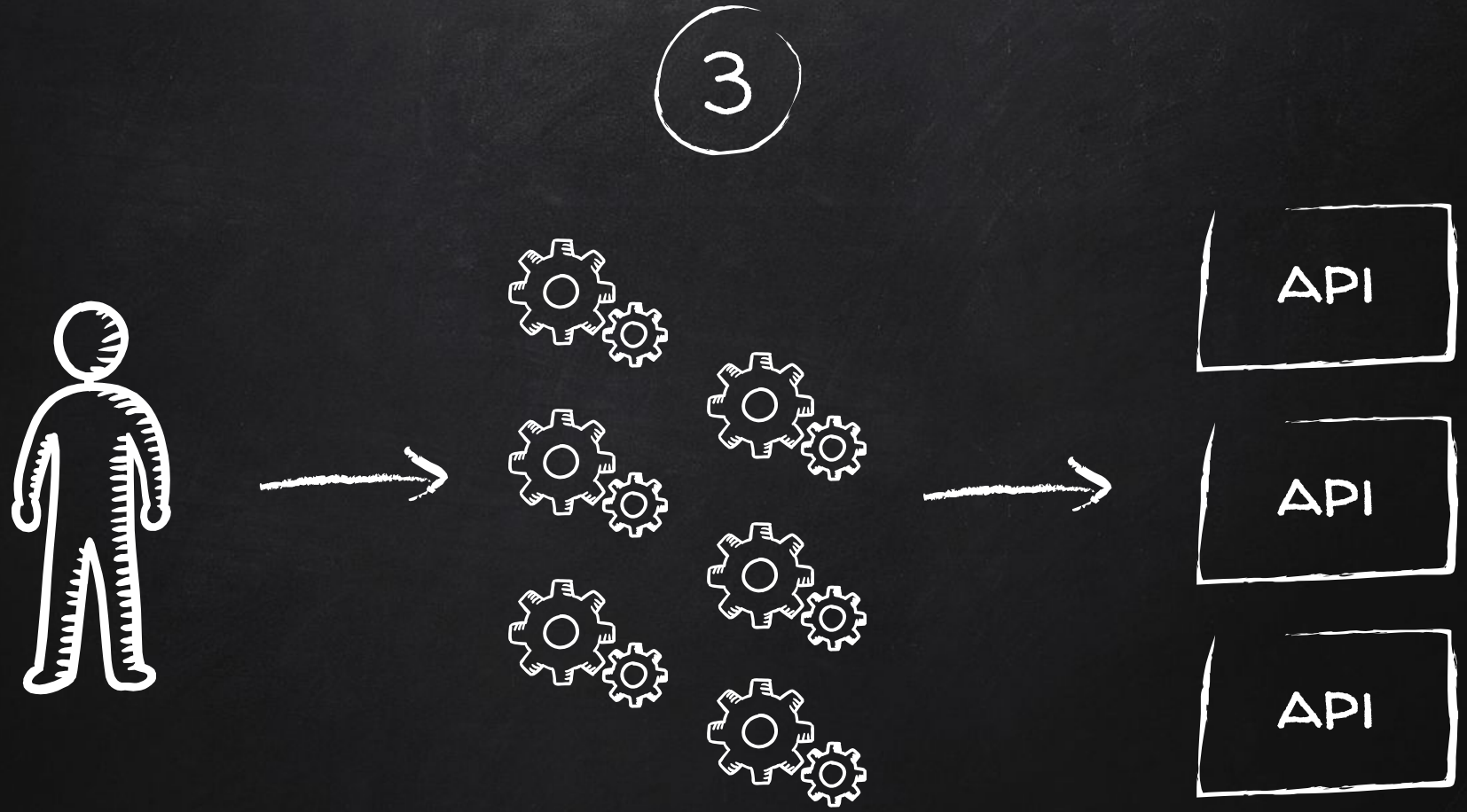

```
var serializer = require("/serious-js-serialiser.js");
serializer.using("someFactory", myFactory);

var myComplexObject = myFactory.buildMeAnObject();
var serial = serializer.serialize(myComplexObject);
// [ now send serial over the network, to file, redis, wherever ]
var clone = serializer.rebuild(serial);
```

2

SCHEDULING


3
w #1
w #2
w #3
w #4

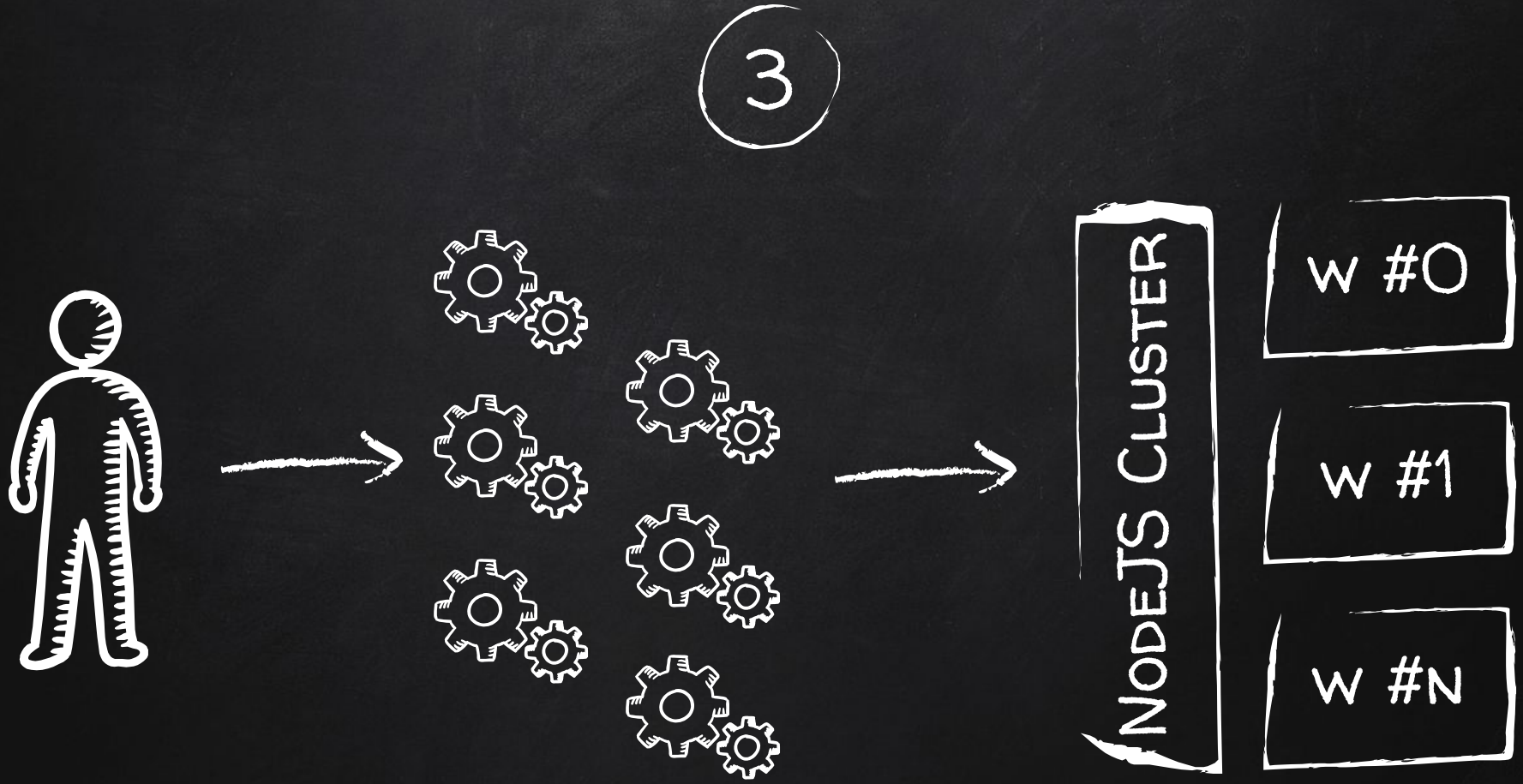

3
API
API
API

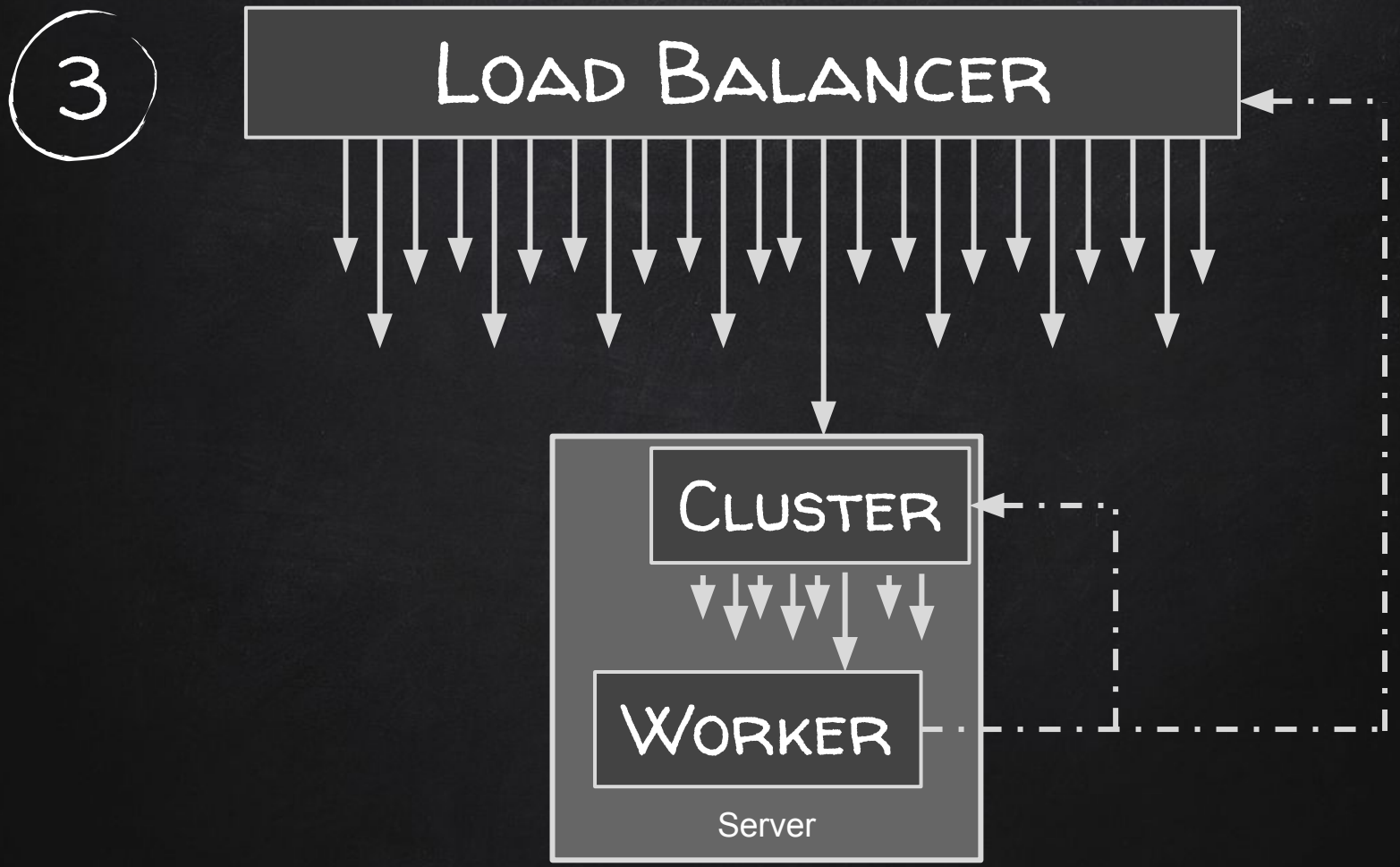
3

The cluster module supports two methods of distributing incoming connections.

The first one (and the default one..), is the round-robin approach, where the master process listens on a port, accepts new connections and distributes them across the workers in a round-robin fashion...

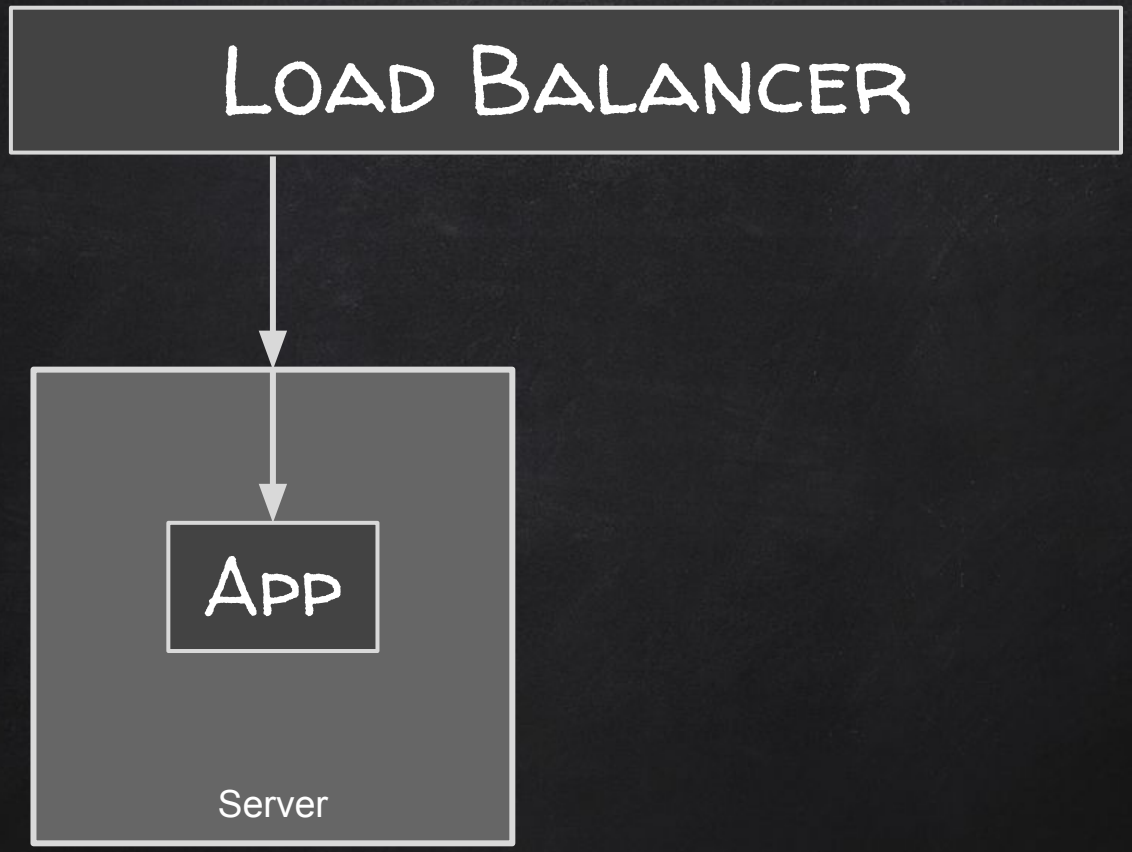
3


3
NodeJS Cluster
W #0
W #1
W #N


3
Load Balancer
Cluster
Worker
Server

DISTRIBUTING COMPLEXITY

4

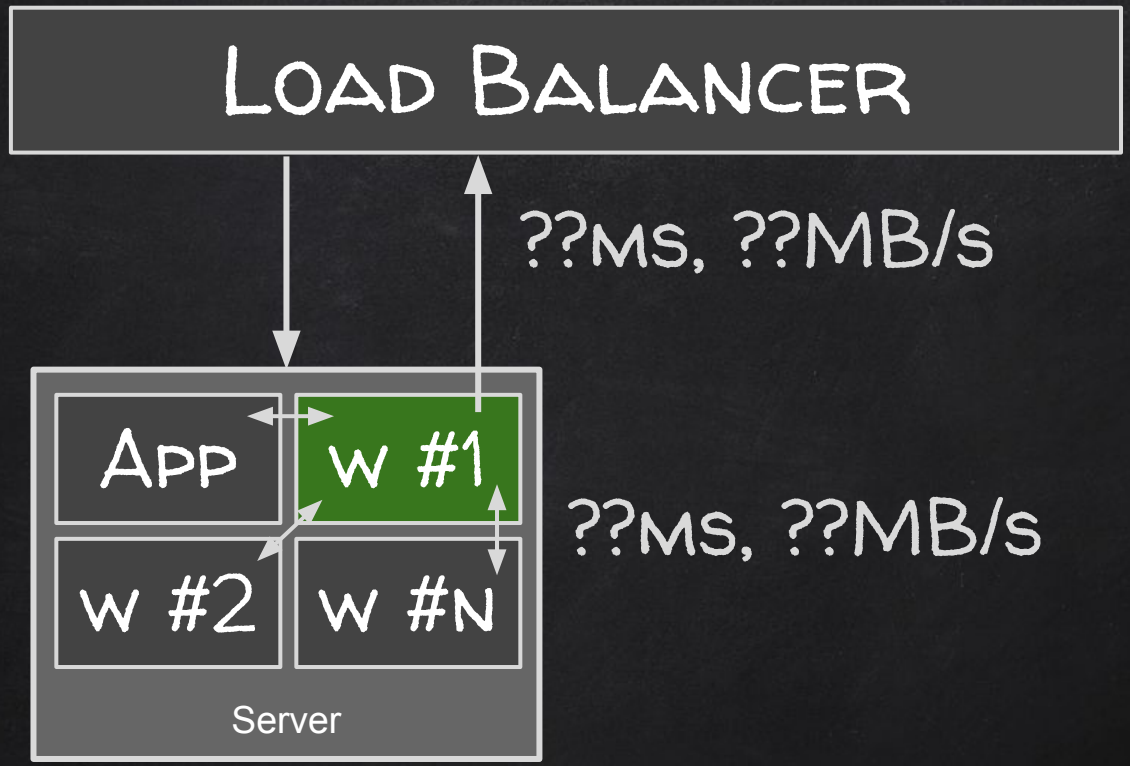



```
var multi = require('./multi.js');
if (multi._isMaster) return;
multi.loadBalancer('localhost');
```

4

Load Balancer

4



4

Load Balancer

Function: mod.foo.bar()

4

Load Balancer

Function: mod.foo.bar()

Async: true

Cost: 1500ms

Blocking: true

4

Load Balancer

Function: mod.foo.bar()

Destination: default

4

Load Balancer

Function: mod.foo.bar()

Destination: local

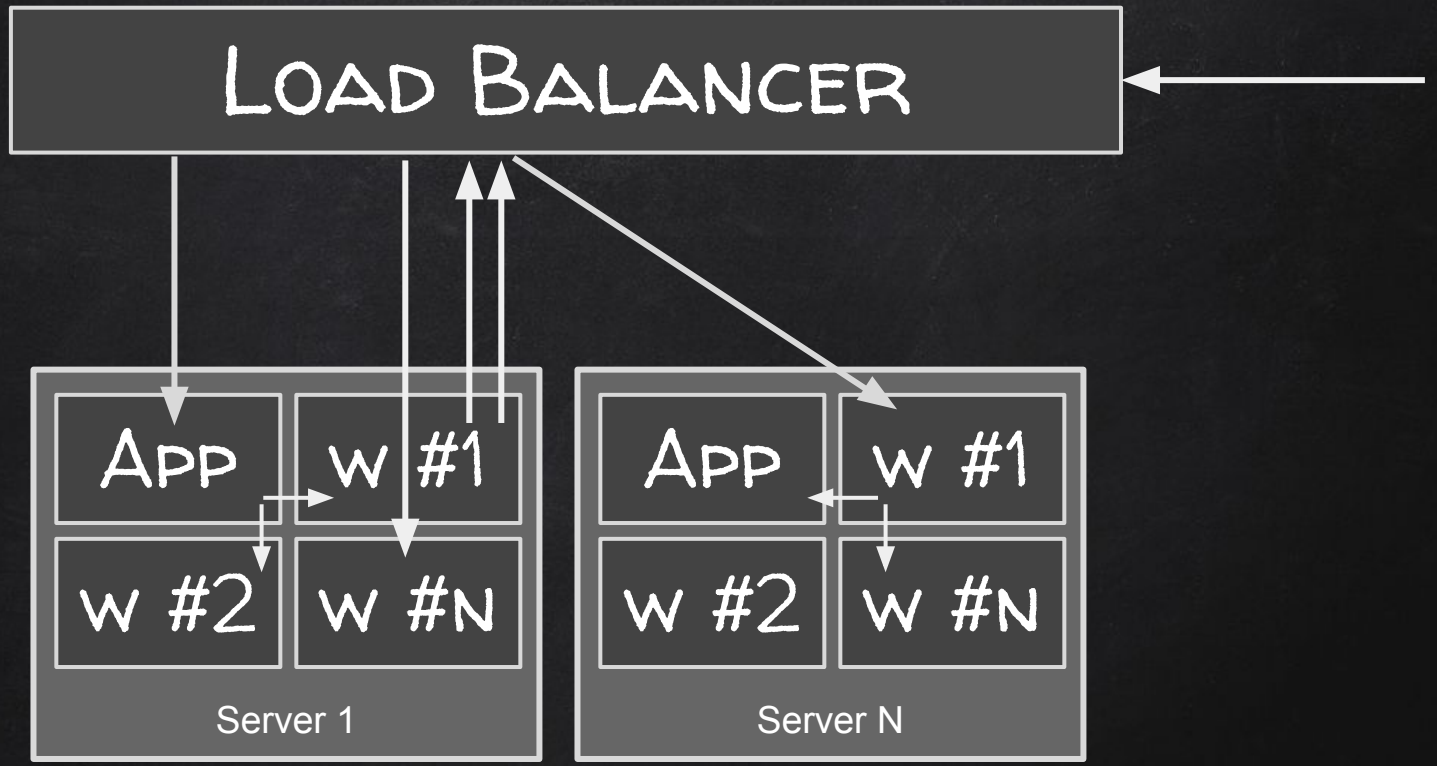
4

Load Balancer

Function: mod.foo.bar()

Destination: remote

4

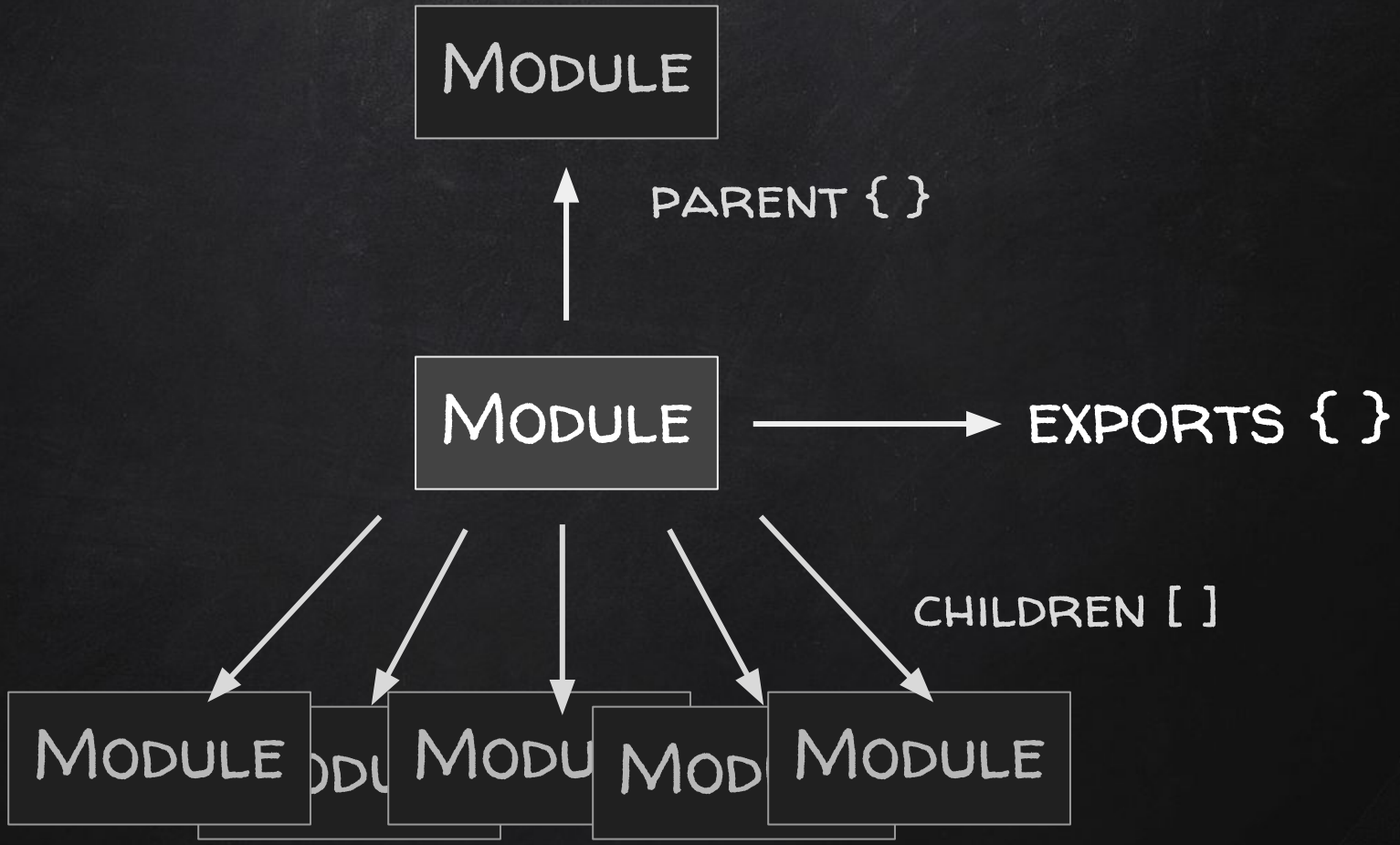



```
module.exports = "Hello world!";
```


```
console.log(module)

> Module {
  exports: {},
  parent: null,
  filename: '/repos/module/mod.js',
  children: []
}
```

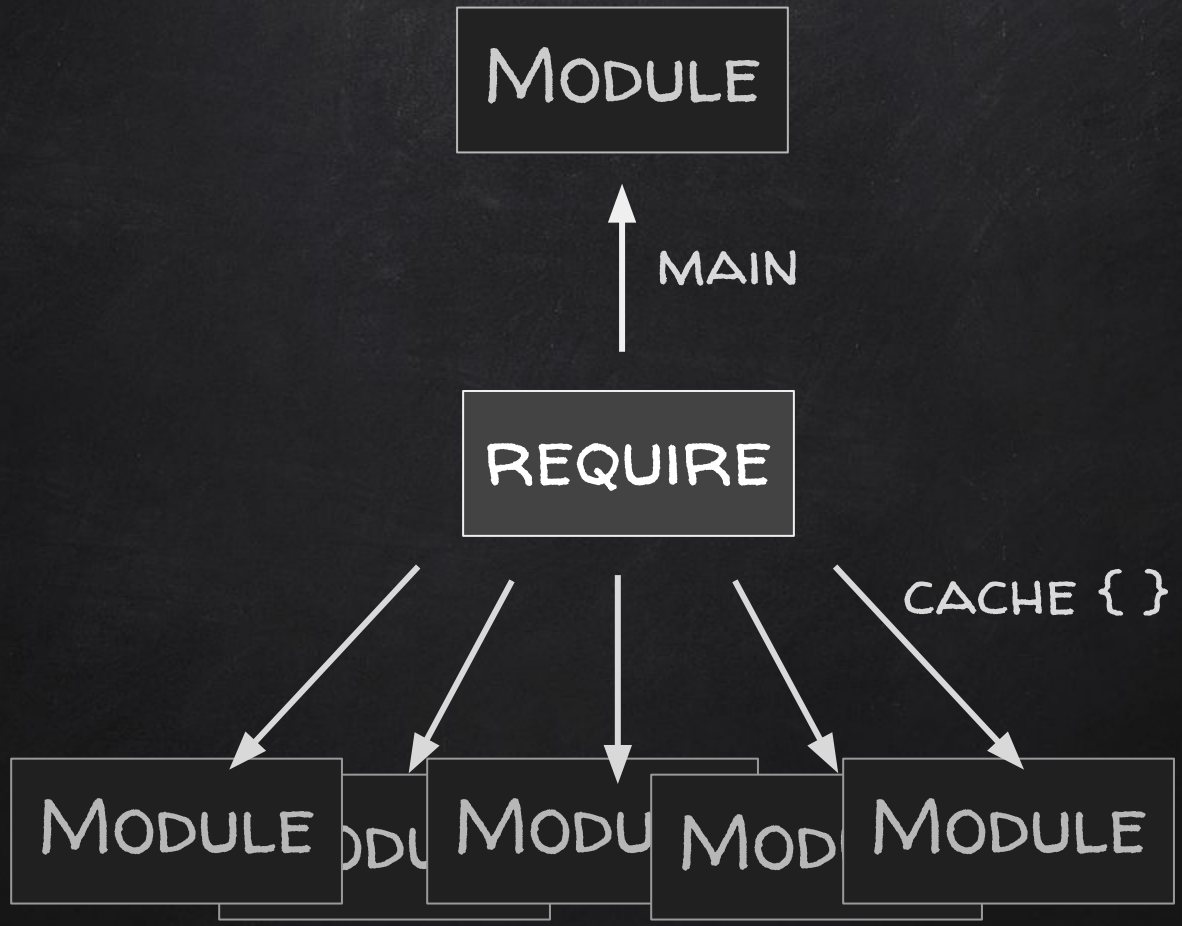
4




```
console.log(require);

> {  [Function: require]
  main: [ Module ],
  cache: {
    '/repos/jsonapi-server/server.js': [ Module ],
    '/repos/jsonapi-server/resources/photos.js': [ Module ],
    '/repos/jsonapi-server/resources/tags.js': [ Module ]
  }
}
```

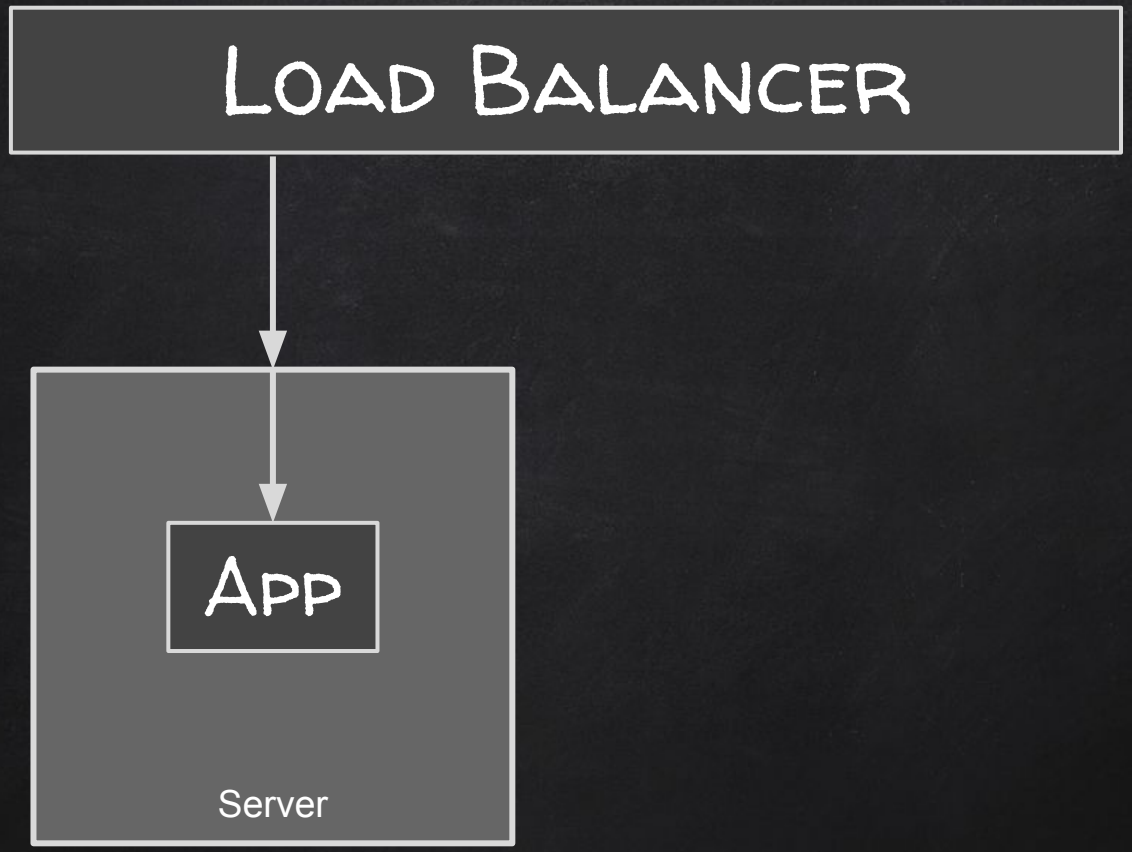
4



https://github.com/holidayextras/hxTracer

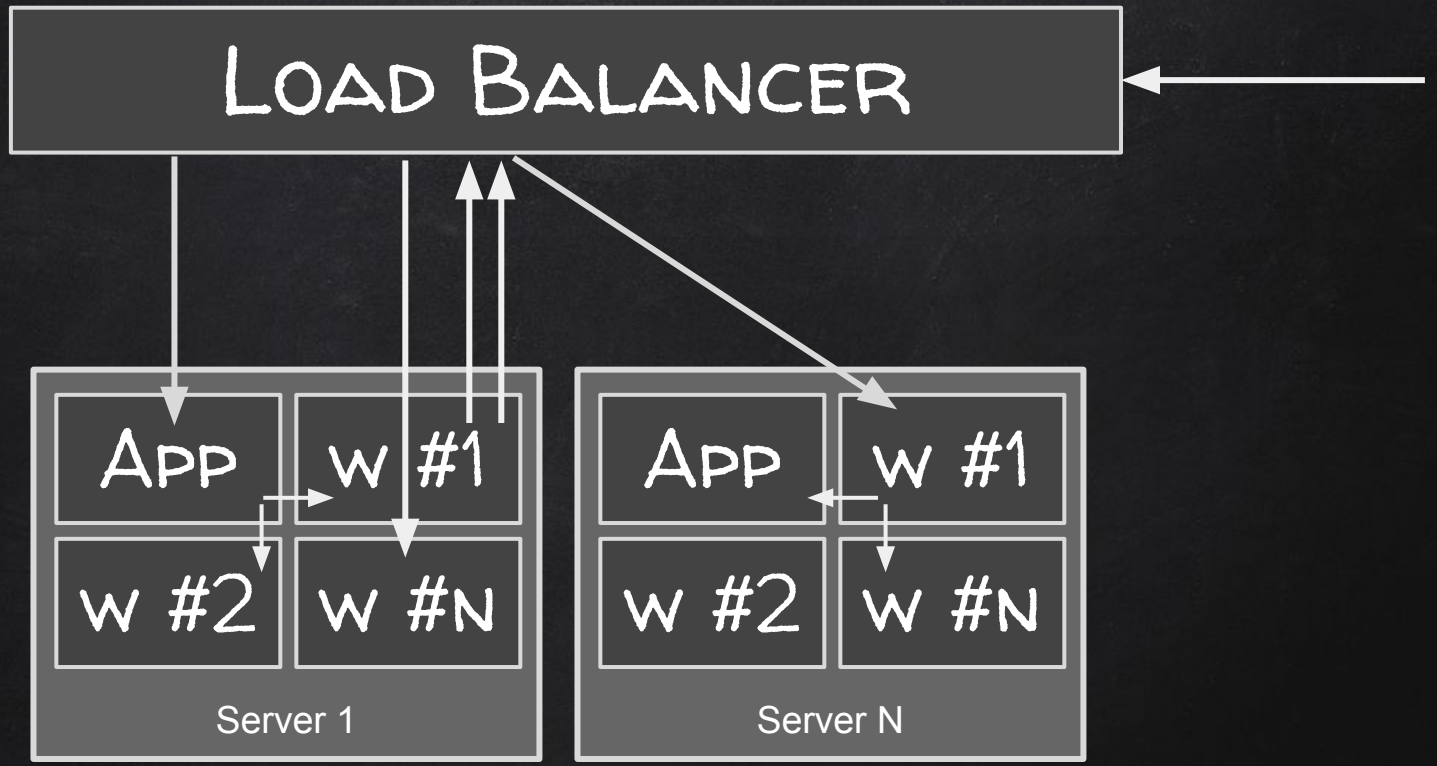
https://github.com/holidayextras/mocha-performance

4




```
var multi = require('./multi.js');
if (multi._isMaster) return;
multi.loadBalancer('localhost');
```

4



Deemmoooooooooo!!!!

https://github.com/theninj4/LNUG-01-16

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash