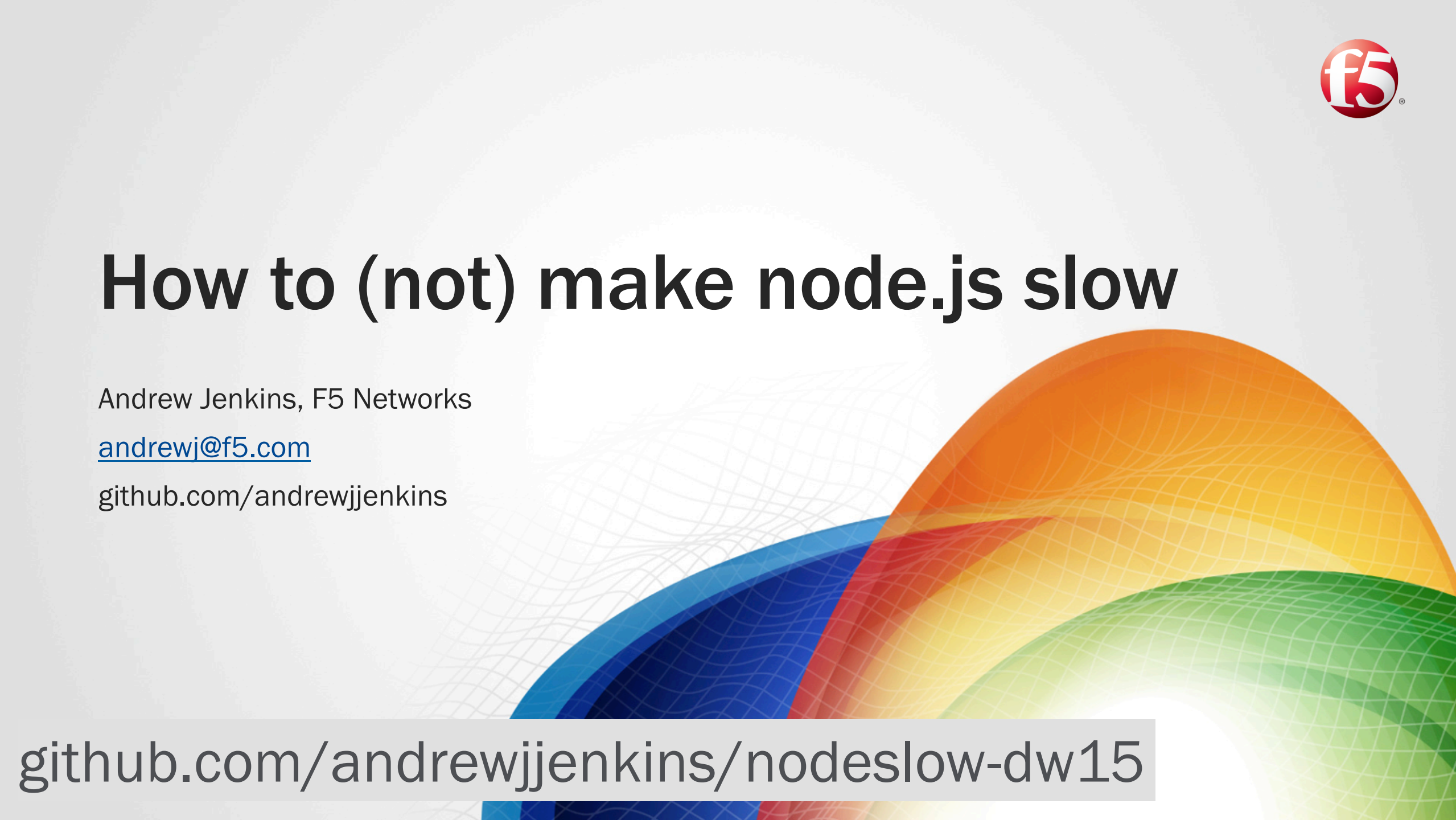


How to (not) make node.js slow

Andrew Jenkins, F5 Networks

andrewj@f5.com

github.com/andrewjjenkins

github.com/andrewjjenkins/nodeslow-dw15

Performance At All Levels

github.com/andrewjjenkins/nodeslow-dw15

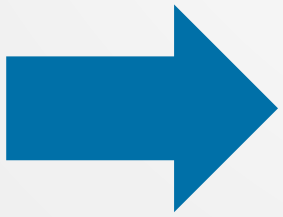
Performance At All Levels

compiler optimization
garbage collection
I/O
entire distributed system

github.com/andrewjjenkins/nodeslow-dw15

Performance At All Levels

compiler optimization
garbage collection
I/O
entire distributed system



TOOLS and TECHNIQUES

Performance At All Levels

compiler optimization

github.com/andrewjjenkins/nodeslow-dw15

Performance At All Levels

compiler optimization
garbage collection

github.com/andrewjjenkins/nodeslow-dw15

Performance At All Levels

compiler optimization
garbage collection
I/O

github.com/andrewjjenkins/nodeslow-dw15

Performance At All Levels

compiler optimization
garbage collection
I/O
entire distributed system

github.com/andrewjjenkins/nodeslow-dw15

TOOLS & TECHNIQUES

github.com/andrewjjenkins/nodeslow-dw15

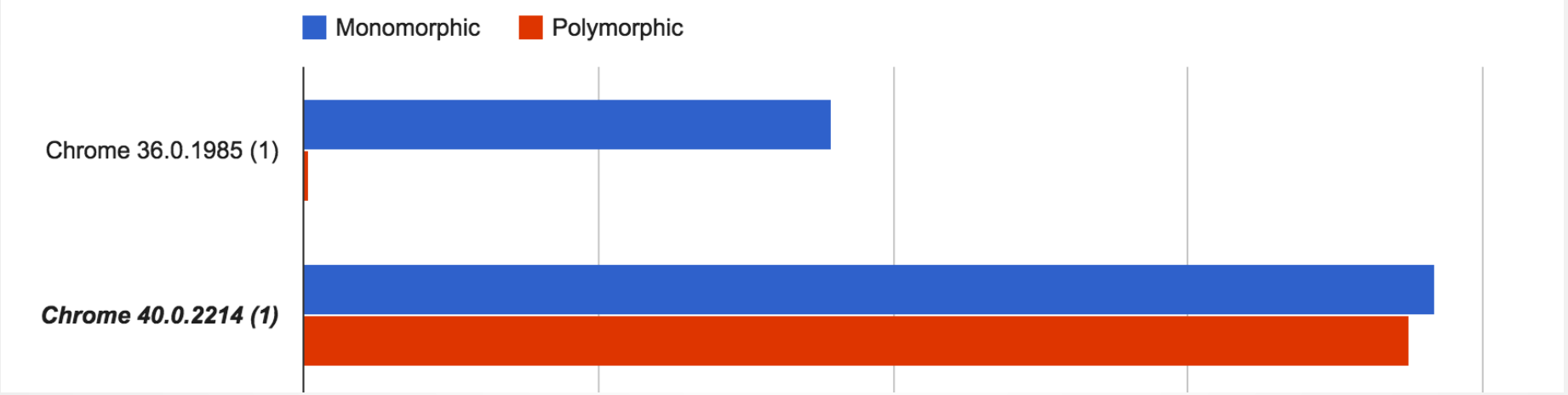
Pull requests if I did something wrong?

Performance At All Levels

"Always call your functions with the same (hidden) types of arguments"

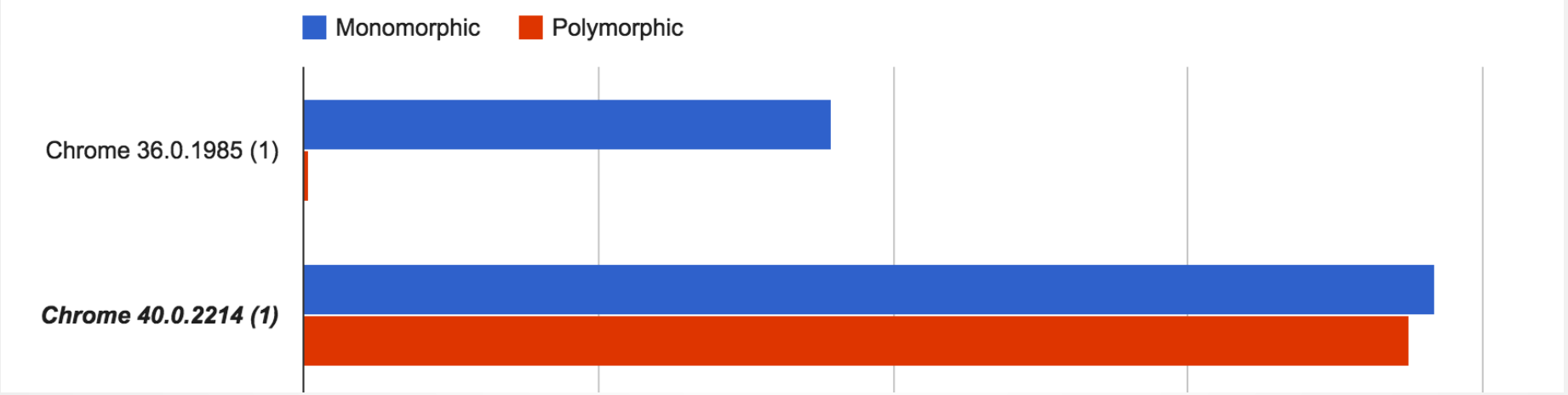
github.com/andrewjjenkins/nodeslow-dw15

"Always call your functions with the same (hidden) types of arguments"



jsperf.com/monomorphic-v8

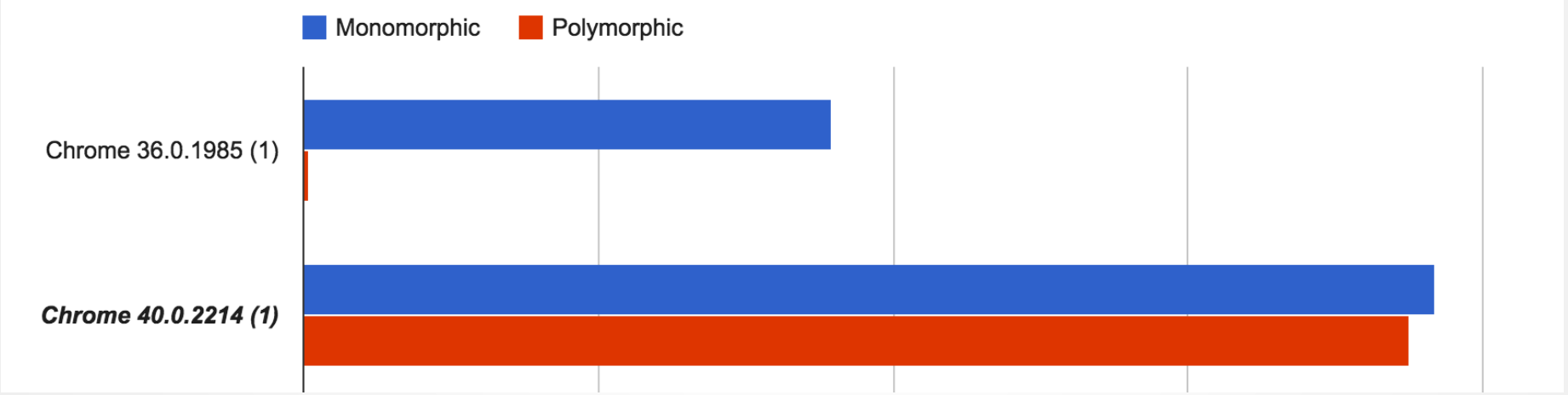
"Always call your functions with the same (hidden) types of arguments"



Mono-Poly-Megamorphism optimization?

jsperf.com/monomorphic-v8

"Always call your functions with the same (hidden) types of arguments"

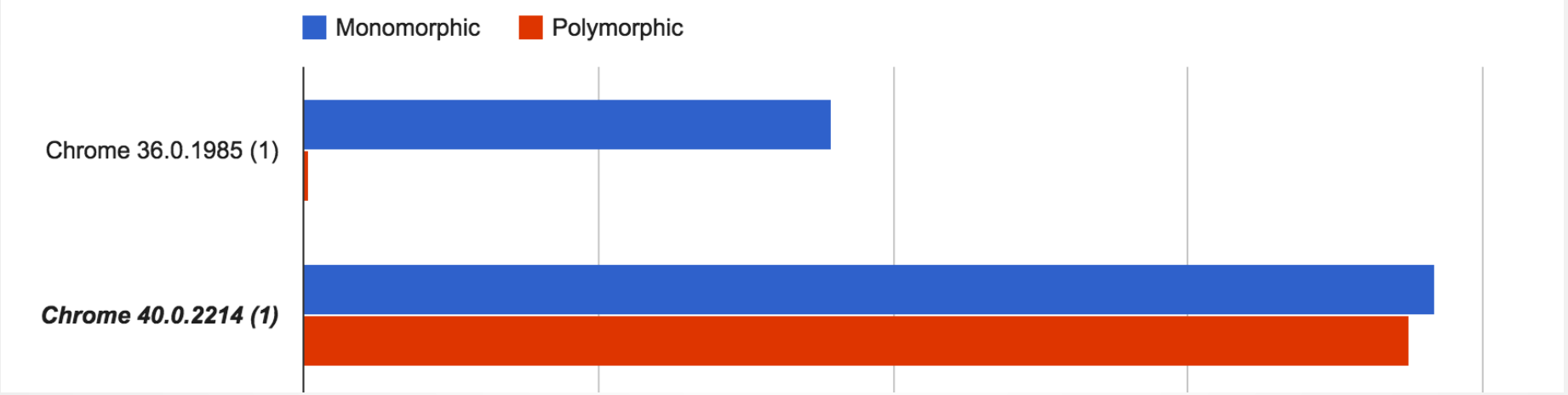


Mono-Poly-Megamorphism optimization?

Benchmark issue?

jsperf.com/monomorphic-v8

"Always call your functions with the same (hidden) types of arguments"



Mono-Poly-Megamorphism optimization?

Benchmark issue?

???

jsperf.com/monomorphic-v8

Performance At All Levels

- compiler optimization
- garbage collection
- I/O
- entire distributed system

TOOL:

```
--trace-(de)opt,
--code-comments,
--print
```

v8 will tell you a lot!

TOOL: `--trace-(de)opt,`
`--code-comments,`
`--print-opt-code`

v8 will tell you a lot!

Extra Credit:

TOOL: `--print-*, IRHydra`

```
npm install benchmark
function doTheAdd(a, b) { return a + b; }
monopoly.add('monomorphic', function monomorphicAdd() {
  doTheAdd(1, 2); doTheAdd(3, 4);
});

monopoly.add('polymorphic', function polymorphicAdd() {
  doTheAdd(1, 2); doTheAdd('a', 'b');
});

monopoly.add('nonemorphic', function nonemorphicAdd() {
  3; 'ab';
});
$ ~/node/node-v0.10.36/bin/node 01-benchmarks/01-monopoly.js
monomorphic x 46,832,958 ops/sec ±0.96% (144 runs sampled)
polymorphic x 18,370,718 ops/sec ±1.45% (149 runs sampled)
nonemorphic x 284,310,468 ops/sec ±12.76% (126 runs sampled)
$ ~/node/node-v0.11.16/bin/node 01-benchmarks/01-monopoly.js
monomorphic x 57,669,735 ops/sec ±0.92% (174 runs sampled)
polymorphic x 59,356,899 ops/sec ±1.47% (151 runs sampled)
nonemorphic x 835,317,996 ops/sec ±0.59% (168 runs sampled)
```

```
node --trace-opt --trace-deopt --code-comments
```

Tell me why

Tell me what's what
in assembly output

```
node --print-opt-code
```

Print the code that the
optimizer produces (v0.11)

```
node --v8-options
```

```
node --trace-opt --trace-deopt --code-comments --print-opt-code
function doTheAdd(a, b) { return a + b; }
monopoly.add('monomorphic', function monomorphicAdd() {
  doTheAdd(1, 2); doTheAdd(3, 4);
});

monopoly.add('polymorphic', function polymorphicAdd() {
  doTheAdd(1, 2); doTheAdd('a', 'b');
});

monopoly.add('nonemorphic', function nonemorphicAdd() {
  3; 'ab';
});
$ ~/node/node-v0.10.36/bin/node 01-benchmarks/01-monopoly.js
monomorphic x 46,832,958 ops/sec ±0.96% (144 runs sampled)
polymorphic x 18,370,718 ops/sec ±1.45% (149 runs sampled)
nonemorphic x 284,310,468 ops/sec ±12.76% (126 runs sampled)
$ ~/node/node-v0.11.16/bin/node 01-benchmarks/01-monopoly.js
monomorphic x 57,669,735 ops/sec ±0.92% (174 runs sampled)
polymorphic x 59,356,899 ops/sec ±1.47% (151 runs sampled)
nonemorphic x 835,317,996 ops/sec ±0.59% (168 runs sampled)
```

0.10.36 / 3.14.5.9

```
[marking doTheAdd 0x3dd24854b9b8 for recompilation,
                  reason: small function, ICs with typeinfo: 1/1 (100%)]
[marking monomorphicAdd 0x3dd2485a2168 for recompilation,
                  reason: small function, ICs with typeinfo: 0/0 (100%)]
[optimizing: monomorphicAdd / 3dd2485a2169 - took 0.064, 0.067, 0.000 ms]
[optimizing: doTheAdd / 3dd24854b9b9 - took 0.022, 0.063, 0.000 ms]


**** DEOPT: doTheAdd at bailout #2, address 0x0, frame size 0
[deoptimizing: begin 0x3dd24854b9b9 doTheAdd @2]
[deoptimizing: end 0x3dd24854b9b9 doTheAdd => node=3, pc=0x2061aaa1d86,
                  state=NO_REGISTERS, alignment=no padding, took 0.022 ms]
[removing optimized code for: doTheAdd]
[marking doTheAdd 0x3dd24854b9b8 for recompilation,
                  reason: small function, ICs with typeinfo: 1/1 (100%)]
[optimizing: doTheAdd / 3dd24854b9b9 - took 0.014, 0.040, 0.000 ms]
[marking polymorphicAdd 0x3dd2485b2388 for recompilation,
                  reason: small function, ICs with typeinfo: 0/0 (100%)]
[optimizing: polymorphicAdd / 3dd2485b2389 - took 0.053, 0.070, 0.000 ms]
```

0.11.16 / 3.28.73

```
[marking 0x.. <JS Fn doTheAdd> for recompilation,
                 reason: small function, ICs with typeinfo: 1/1 (100%),
                 generic ICs: 0/1 (0%)]
[marking 0x.. <JS Fn monomorphicAdd> for recompilation,
                 reason: small function, ICs with typeinfo: 2/2 (100%),
                 generic ICs: 0/2 (0%)]
[optimizing 0x.. <JS Fn monomorphicAdd> - took 0.068, 0.122, 0.051 ms]
[optimizing 0x.. <JS Fn doTheAdd> - took 0.016, 0.060, 0.029 ms]

[deoptimizing (DEOPT eager): begin 0x.. doTheAdd (opt #33) @1, FP to SP
delta: 16]
[evicting entry from optimizing code map (notify deoptimized) for
0x3f5465f235f9 <SharedFunctionInfo doTheAdd>]
[marking 0x.. <JS Fn polymorphicAdd> for recompilation,
                 reason: small function, ICs with typeinfo: 2/2 (100%),
                 generic ICs: 0/2 (0%)]
[optimizing 0x.. <JS Fn polymorphicAdd> - took 0.060, 0.107, 0.046 ms]
```

0.11.16 / 3.28.73

```
... STACK BOILERPLATE ...
<@0,#0> ---- B0 ----
REX.W movq rax,[rbp-0x8]      ;; context
REX.W movq [rbp-0x18],rax     ;; gap
<@8,#8> ---- B1 ----
REX.W movq rsi,rax            ;; gap
REX.W cmpq rsp,[r13+0x798]    ;; stack-check
jnc 35  (0x1f040178bfa3)
call StackCheck  (0x1f0401633820)  ;; BUILTIN
REX.W movq rax,[rbp-0x18]     ;; lazy-bailout, gap
REX.W movq rax,[rax+0x2f]     ;; load-context-slot
<@16,#15> check-value
REX.W movq r10,0x15d9421631c9
    ;; object: 0x15d9421631c9 <JS Fn doTheAdd>
REX.W cmpq rax,r10
jnz 79  (0x1f040178bfcf)
<@20,#27> ---- B2, B3 ----
<@26,#4> constant-t
REX.W movq rax,0x19859ea04121    ;; debug: position 226
                                 ;; object: 0x19859ea04121 <undefined>
<@28,#44> return ..
```

Performance At All Levels

- compiler optimization
- garbage collection
- I/O
- entire distributed system

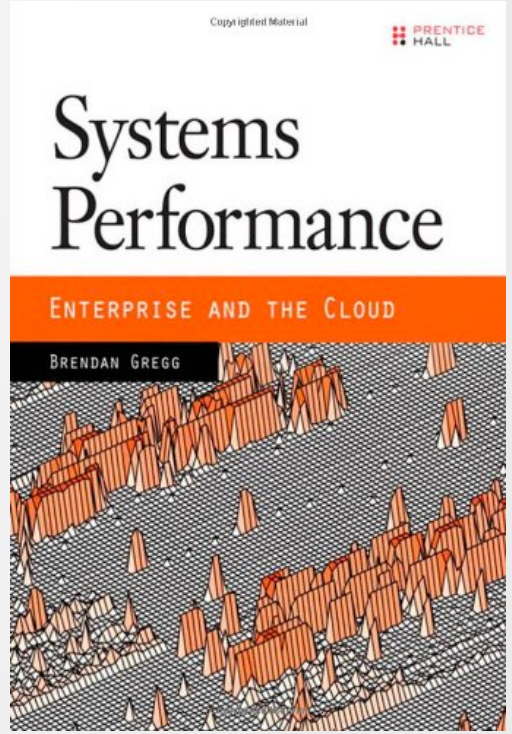
TECHNIQUE: Experts

Vyacheslav Egorov, mrale.ph

Thorsten Lorenz, github.com/thlorenz/v8-perf

Trevor Norris, blog.trevnorris.com

Brendan Gregg, Systems Performance



Performance At All Levels

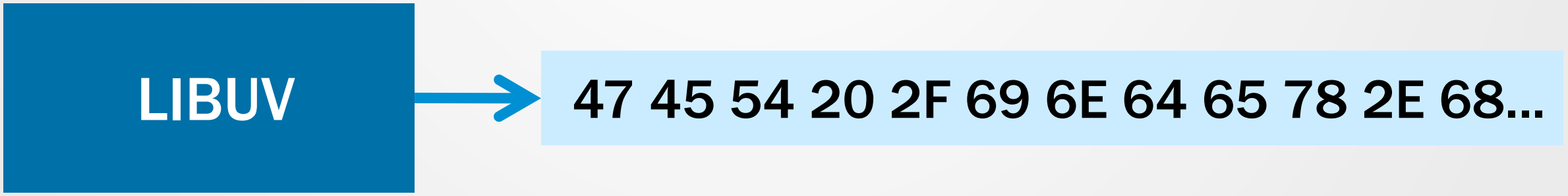
compiler optimization

garbage collection

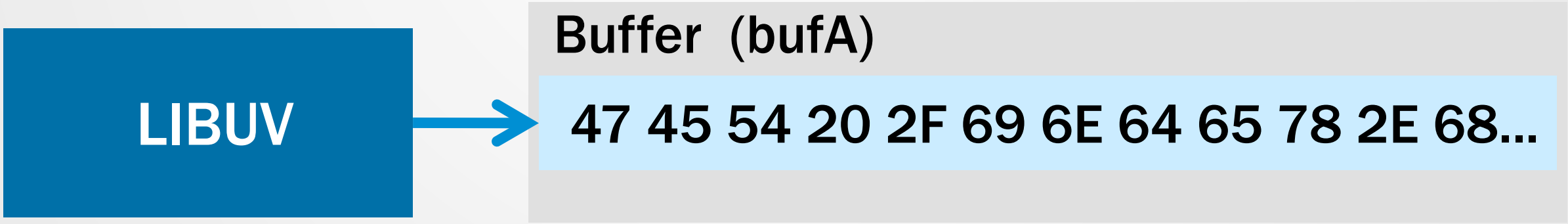
I/O

entire distributed system

Buffers


LIBUV
47 45 54 20 2F 69 6E 64 65 78 2E 68...

SetIndexedPropertiesToExternalArrayData



So your node.js program is humming along...

So your node.js program is humming along...

But *some* transactions double-processed *somewhere*

So your node.js program is humming along...

But *some* transactions double-processed *somewhere*

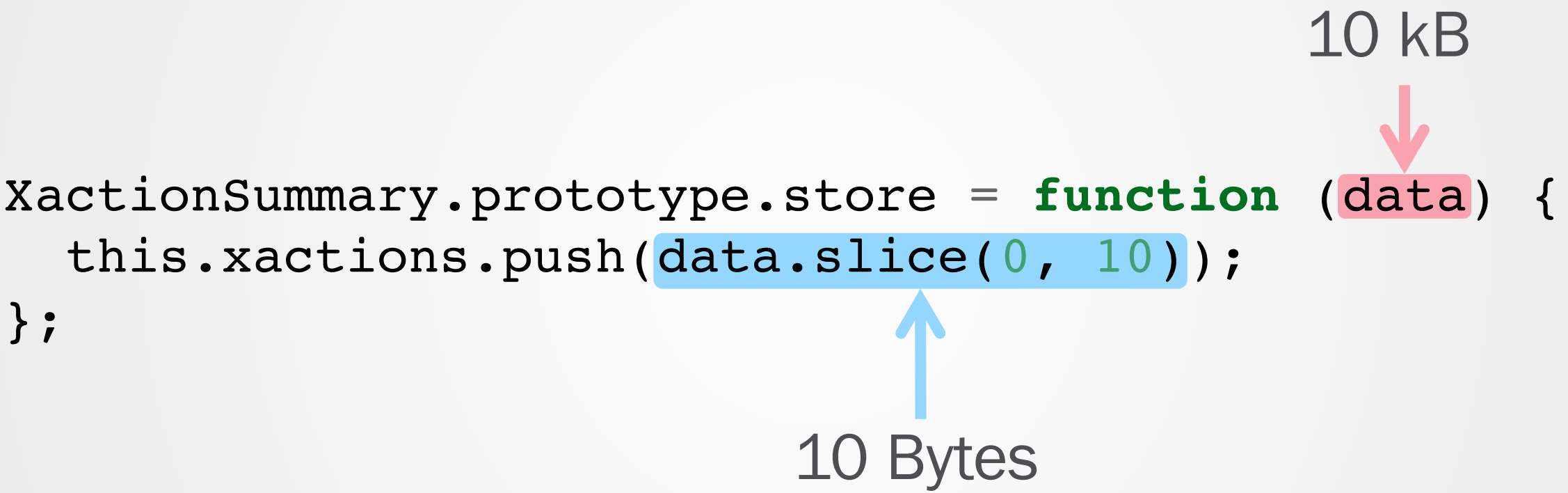
...Let's buffer some summary of the data

```
XactionSummary.prototype.store = function (data) {
  this.xactions.push(data.slice(0, 10));
};
```

Request: 10kB
Summary: first 10 bytes

Slice off 10 bytes, store. Dump in the middle of the night.

100,000 reqs/day
10 * 100,000 ≈ 1MB. Easy!


10 kB
XactionSummary.prototype.store = function (data) {
this.xactions.push(data.slice(0, 10));
};
10 Bytes

Friday, 11:32 pm

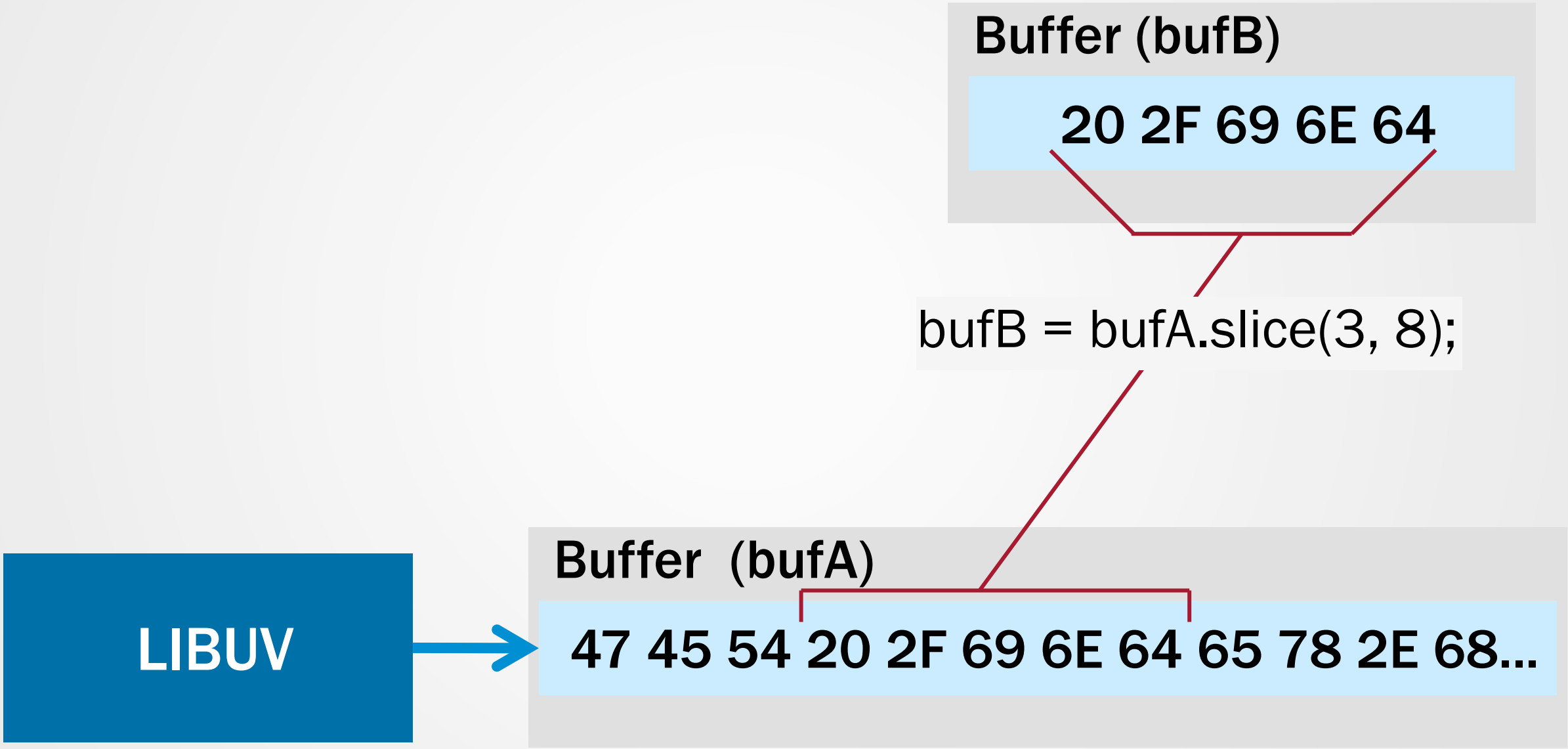
...ring, ring

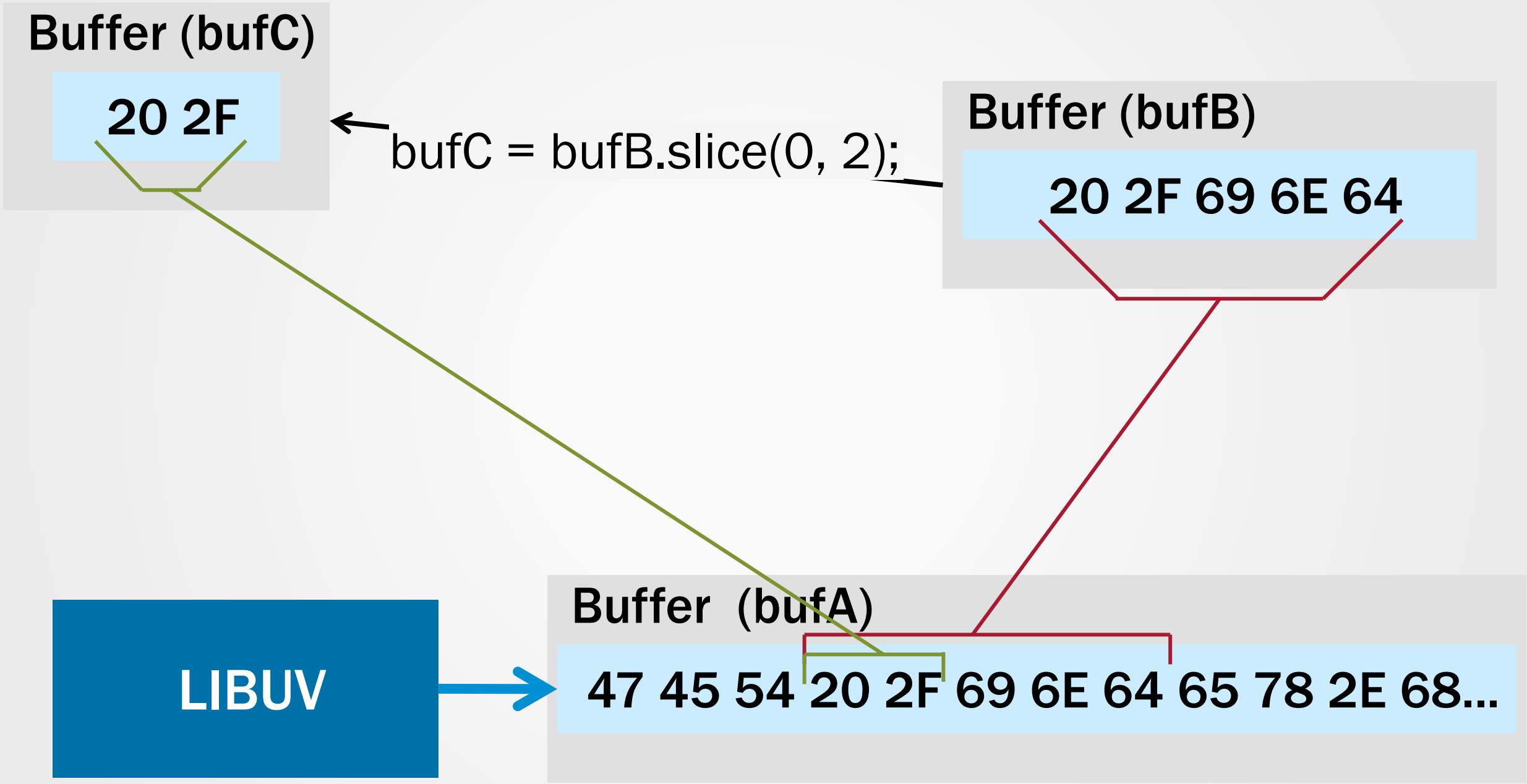
Hello?

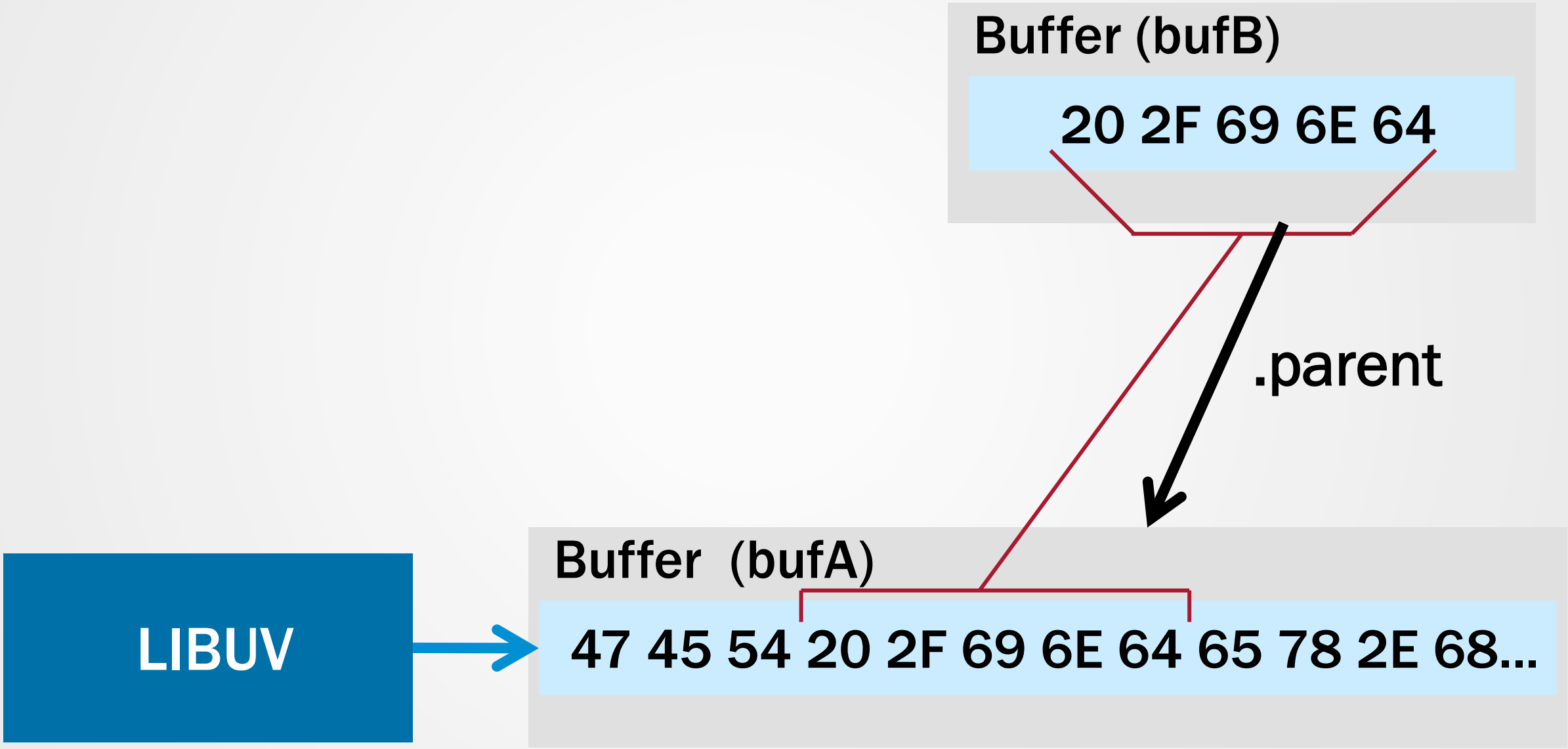
node ate up all the memory
got really slow, and then crashed.

time -l node --trace-gc --expose-gc 02-buffers/buffers.js

```
100000 buffers, total 1000000 bytes (avg: 10)
[83816]     2702 ms: Mark-sweep 20.6 (55.0) -> 20.2 (55.0) MB, 27 ms
                     [gc extension] [GC in old space requested].
            2.82 real         1.81 user         1.01 sys
      1093216  maximum resident set size
         8530  average shared memory size
         1509  average unshared data size
          125  average unshared stack size
          ...  ...
```


Buffer (bufB)
20 2F 69 6E 64
bufB = bufA.slice(3, 8);
Buffer (bufA)
47 45 54 20 2F 69 6E 64 65 78 2E 68...
LIBUV


Buffer (bufC)
20 2F
bufC = bufB.slice(0, 2);
Buffer (bufB)
20 2F 69 6E 64
LIBUV
Buffer (bufA)
47 45 54 20 2F 69 6E 64 65 78 2E 68...


Buffer (bufB)
20 2F 69 6E 64
.parent
Buffer (bufA)
47 45 54 20 2F 69 6E 64 65 78 2E 68...
LIBUV

```
XactionSummary.prototype.store = function (data) {
    var summaryBuf = new Buffer(10);
    data.copy(summaryBuf, 0, 10);
    this.xactions.push(summaryBuf);
};
```

```
STRATEGY="copy" time -l node --trace-gc --expose-gc \
   02-buffers/buffers.js

100000 buffers, total 1000000 bytes (avg: 10)
[84250]     1977 ms: Mark-sweep 23.9 (52.0) -> 19.0 (54.0) MB,
         15 / 33 ms [gc extension] [GC in old space requested].
           2.00 real         1.56 user         0.44 sys
      160020  maximum resident set size
        8665  average shared memory size
        1537  average unshared data size
         127  average unshared stack size
         ...  ...
```

Hmmmm... what about new Buffer(10) ?

xactions[0]

20 2F 6E 64

xactions[1]

21 45 .. .. 6E 34

xactions[2]

27 20 .. .. 61 64

xactions[3]

61 2A 60 33

xactions[4]

61 29 .. .. 60 33

```
function XactionSummary() {
  this.xactions = [];
  this.buf = new Buffer(10 * requestsToProcess);
  this.bufCursor = 0;
}
XactionSummary.prototype.store = function (data) {
  var summaryBuf = this.buf.slice(this.bufCursor,
                                  this.bufCursor+10);
  data.copy(summaryBuf, 0, 10);
  this.bufCursor += 10;
  this.xactions.push(summaryBuf);
};
```

```
STRATEGY="copyPrealloc" time -l node --trace-gc --expose-gc \
  02-buffers/buffers.js

100000 buffers, total 10 bytes (avg: 0.0001)
[85019]     1985 ms: Mark-sweep 23.9 (52.0) -> 19.0 (54.0) MB,
              15 / 33 ms [gc extension] [GC in old space requested].
            2.01 real         1.69 user         0.31 sys
       158076  maximum resident set size
         8734  average shared memory size
         1550  average unshared data size
          128  average unshared stack size
        37063  page reclaims
          ...  ...
```

```
Buffer.poolSize = 8 * 1024;
```

```
function XactionSummary() {
  this.buf = new Buffer(10 * requestsToProcess);
  this.bufCursor = 0;
}
XactionSummary.prototype.store = function (data) {
  data.copy(this.buf, this.bufCursor,
                      this.bufCursor + 10);
  this.bufCursor += 10;
};
```

```
STRATEGY="copyPreallocNoSlice" time -l node --trace-gc \
  --expose-gc 02-buffers/buffers.js

100000 buffers, total 1000000 bytes (avg: 10)
[86478]     1914 ms: Mark-sweep 2.2 (37.0) -> 1.8 (37.0) MB,
                1 ms [gc extension] [GC in old space requested].
            1.93 real         1.52 user         0.41 sys
          150212  maximum resident set size
            8735  average shared memory size
            1544  average unshared data size
             128  average unshared stack size
             ...  ...
```

Performance At All Levels

compiler optimization

garbage collection

I/O

entire distributed system

Example: Async

Alice connects to Bob

Notification

Buddy List

Email Alice & Bob

Email one-hop neighbors

Email two-hop neighbors

```
function handleNewConnection(userA, userB, cb) {
  notifyPhoneNewConnection(userB, userA.name, function (err) {
    if (err) return cb(err);
    addToBuddyList(userB, userA.name, function(err) {
      if (err) return cb(err);
      emailNewConnection(userB, userA.name, function (err) {
        if (err) return cb(err);
        emailNewConnection(userA, userB.name, function (err) {
          findOneHopNeighbors(userA.name, userB.name, function (err, neighbors) {
            if (err) return cb(err);
            function emailNextNeighbor(i) {
              if (i < neighbors.length) {
                emailCommonNeighbor(neighbors[i].name, userA.name, userB.name, function (err) {
                  if (err) return cb(err);
                  emailNextNeighbor(i + 1);
                });
              } else {
                findTwoHopNeighbors(userA.name, userB.name, function (err) { /* ... */ });
              }
            }
            emailNextNeighbor(0);
          });
        });
      });
    });
  });
}
```

Control Flow Libraries

async

```
async.series([notifyPhoneNewConnection, ... ], cb);
```

github.com/caolan/async (npm install async)

promises

```
notifyPhoneNewConnection.then(addToBuddyList).then(...);
```

github.com/petkaantonov/bluebird (npm install bluebird)

generators

```
yield notifyPhoneNewConnection; yield addToBuddyList; ...
```

github.com/strongloop/example-generators (Need `node --harmony`)

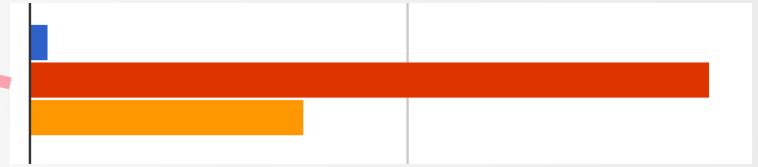
fibers

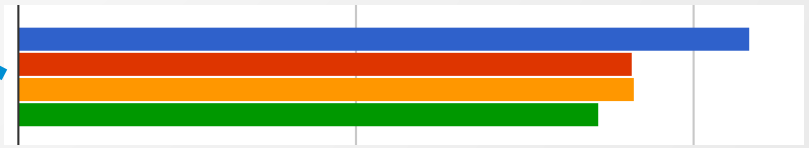
```
notifyPhoneNewConnection(); addToBuddyList();
```

github.com/laverdet/node-fibers (+ wrapper, req's compile) (npm install fibers)

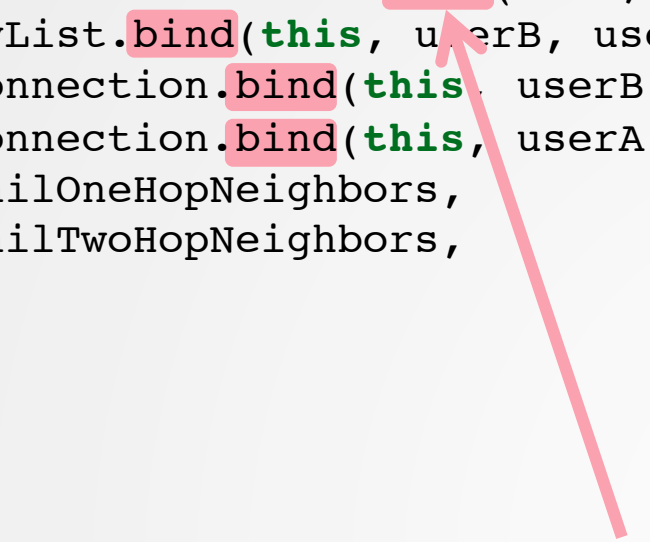
```
function handleNewConnection(userA, userB, cb) {
  async.series([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.series(emailers, cb);
  }
  /* ... */
}
```

```
function handleNewConnection(userA, userB, cb) {
  async.series([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(this, userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.series(emailers, cb);
  }
  /* ... */
}
```





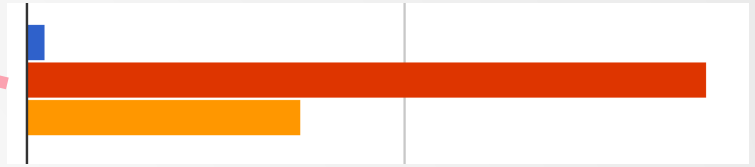
```
function handleNewConnection(userA, userB, cb) {
  async.series([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);
```



```
function notifyPhoneNewConnectionBinder(userB, userAName) {
  return function (cb) {
    notifyPhoneNewConnection(userB, userAName, cb);
  }
}
```

```
}
```

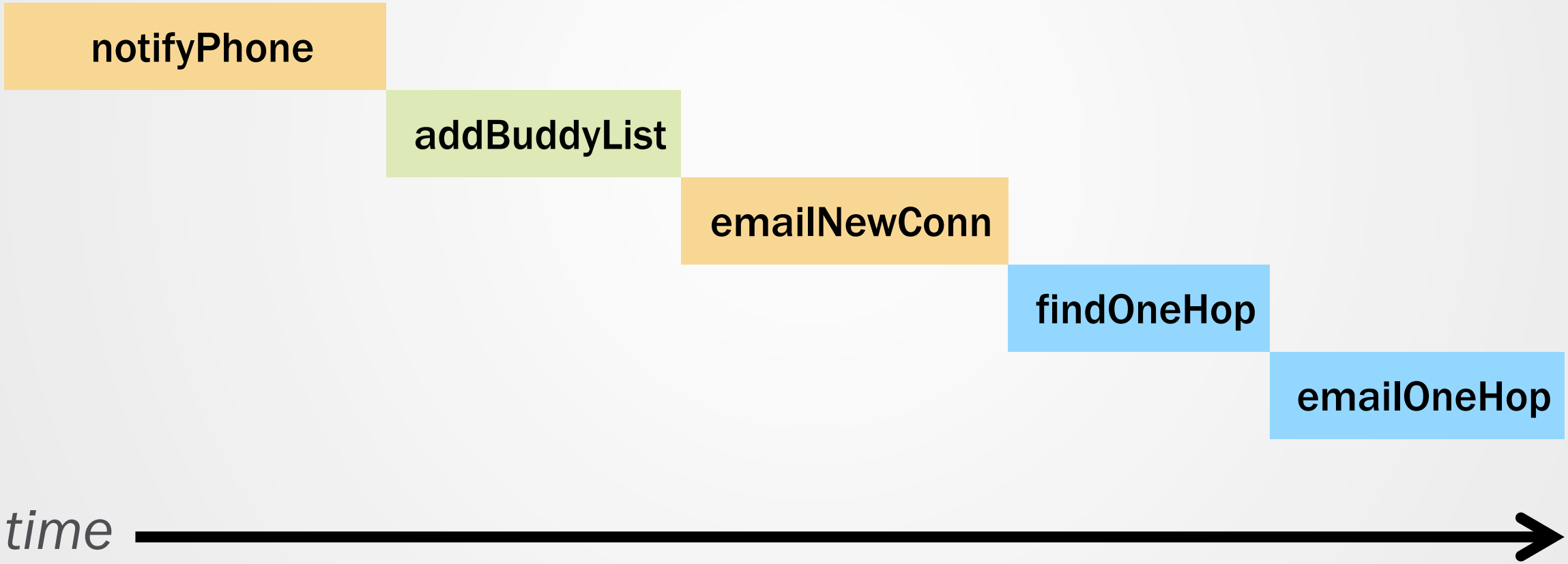
```
function handleNewConnection(userA, userB, cb) {
  async.series([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.series(emailers, cb);
  }
  /* ... */
}
```



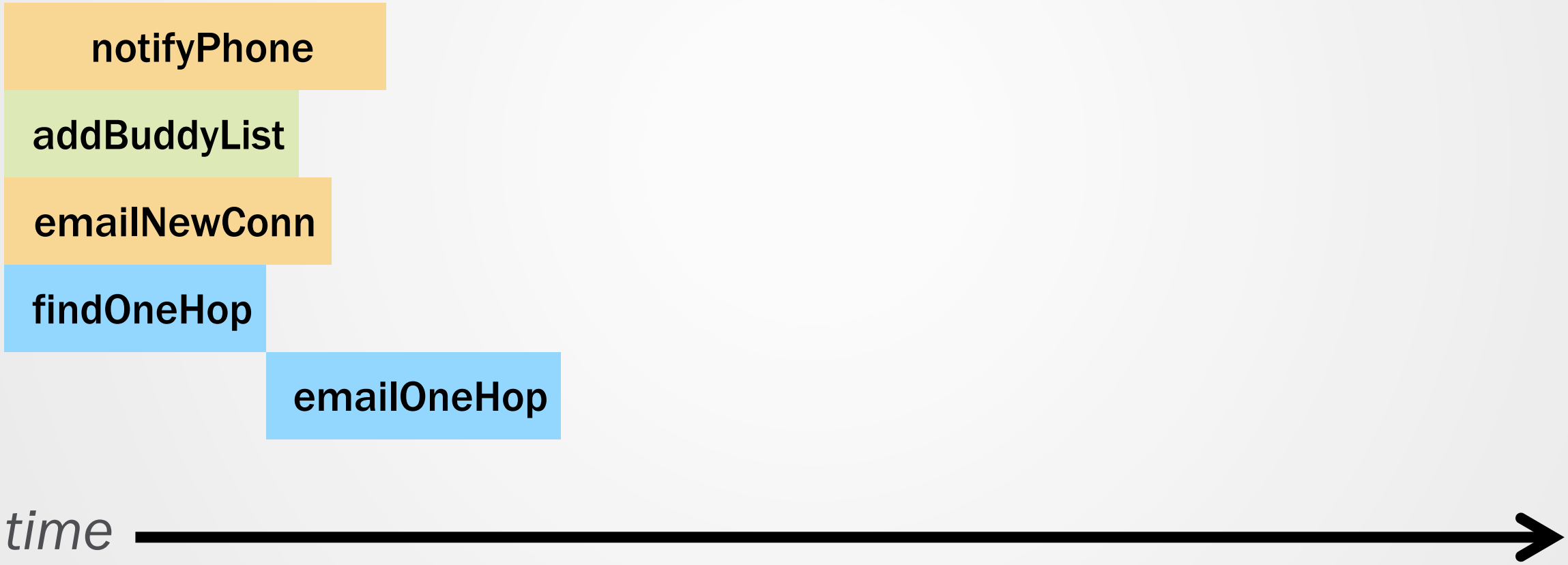
```
function handleNewConnection(userA, userB, cb) {
  function notifyMyPhoneNewConnection(cb) {
    notifyPhoneNewConnection(userB, userA.name, cb);
  }
  async.series([
    notifyMyPhoneNewConnection,
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.series(emailers, cb);
  }
  /* ... */
}
```

Parallel?

Series


notifyPhone
addBuddyList
emailNewConn
findOneHop
emailOneHop
time

Parallel


notifyPhone
addBuddyList
emailNewConn
findOneHop
emailOneHop
time

```
function handleNewConnection(userA, userB, cb) {
  async.series([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.series(emailers, cb);
  }
  /* ... */
}
```

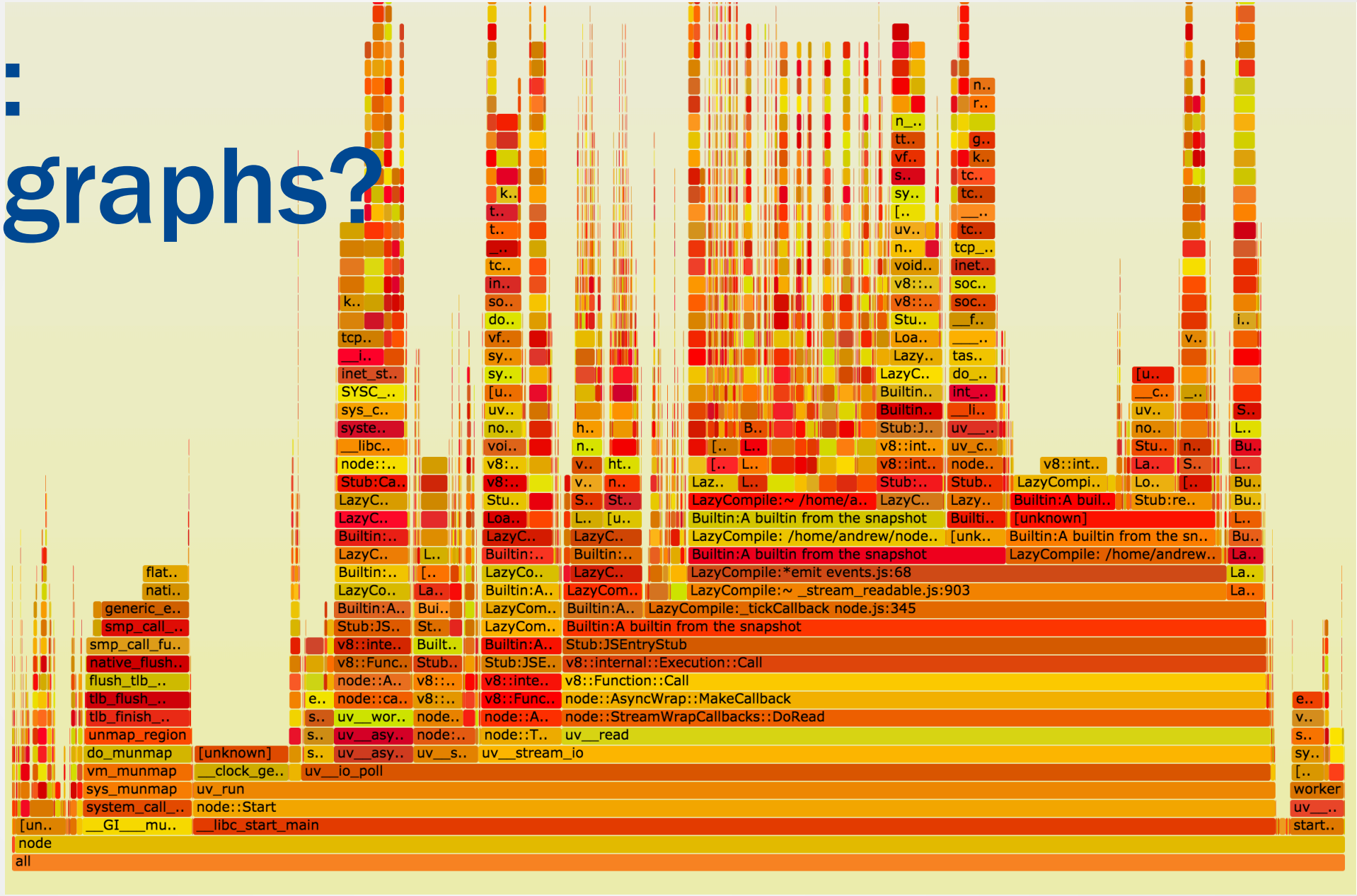
```
function handleNewConnection(userA, userB, cb) {
  async.parallel([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.parallel(emailers, cb);
  }
  /* ... */
}
```

```
function handleNewConnection(userA, userB, cb) {
  async.parallel([
    notifyPhoneNewConnection.bind(this, userB, userA.name),
    addToBuddyList.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userB, userA.name),
    emailNewConnection.bind(this, userA, userB.name),
    findAndEmailOneHopNeighbors,
    findAndEmailTwoHopNeighbors,
  ], cb);

  function findAndEmailOneHopNeighbors(cb) {
    async.waterfall([
      findOneHopNeighbors.bind(userA.name, userB.name),
      emailCommonNeighbors,
    ], cb);
  }
  function emailCommonNeighbors(neighbors, cb) {
    var emailers = [];
    for (var i = 0; i < neighbors.length; i++) {
      emailers.push(emailCommonNeighbor.bind(this, neighbors[i].name, userA.name, user));
    }
    async.parallelLimit(emailers, 10, cb);
  }
  /* ... */
}
```

Performance At All Levels

compiler optimization
garbage collection
I/O
entire distributed system

Bonus: Flamegraphs?



TOOLS & TECHNIQUES

github.com/andrewjjenkins/nodeslow-dw15

Andrew Jenkins

andrewj@f5.com , github.com/andrewjjenkins