THE UNIVERSITY OF BRITISH COLUMBIA

Julien Gascon-Samson
Kumseok Jung
Shivanshu Goyal
Armin Rezaiean-Asel
Karthik Pattabiraman

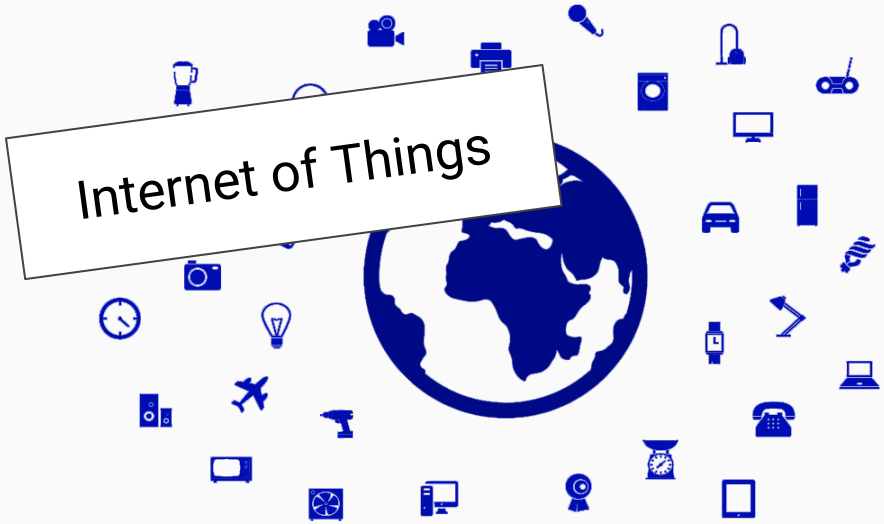
ThingsMigrate

Platform-independent and stateful migration of JavaScript programs

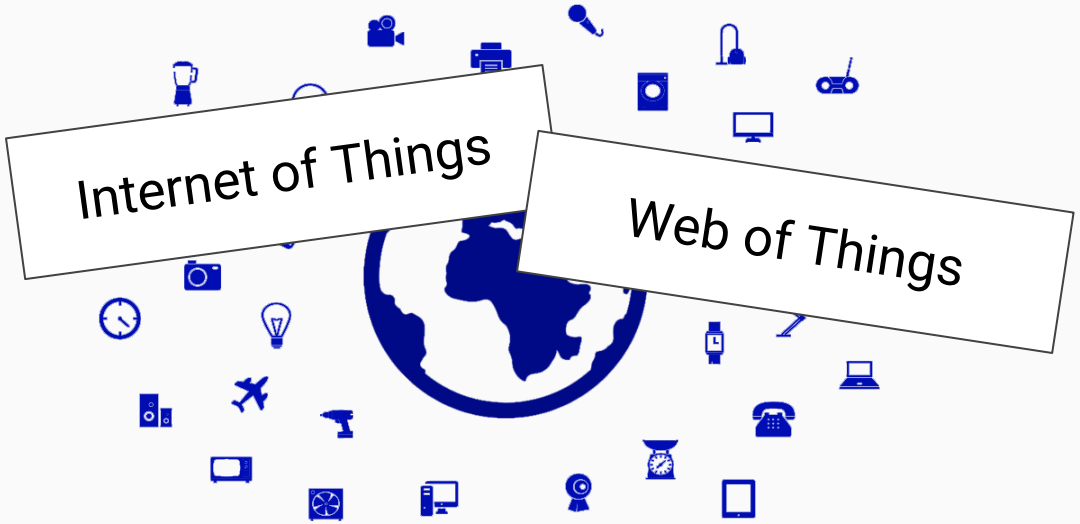
ICICS

World of computing in 2018

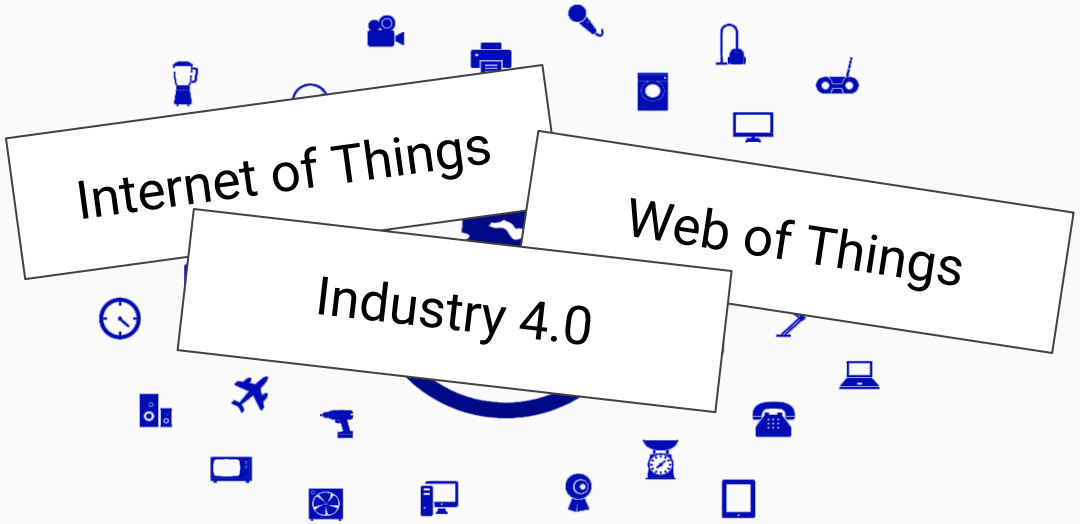
World of computing in 2018



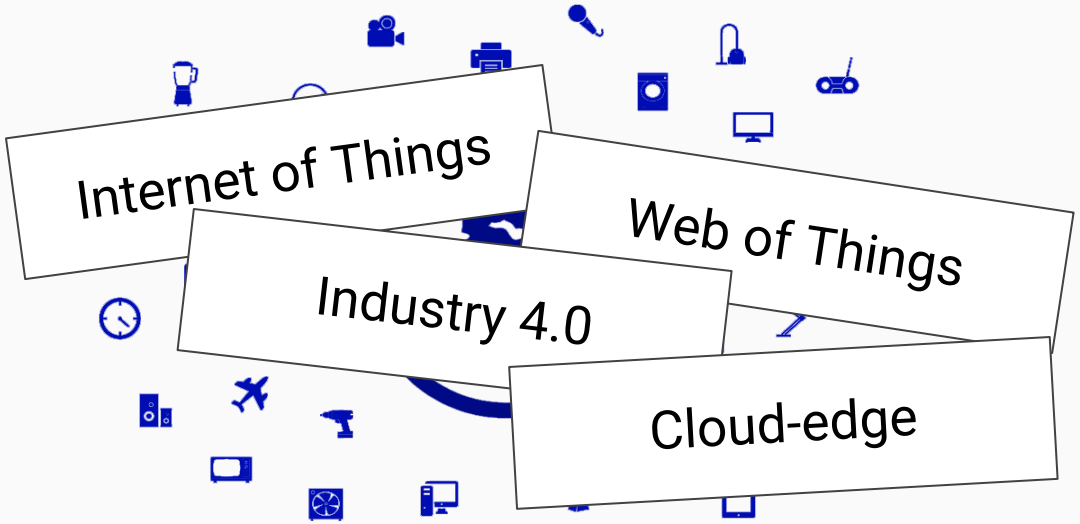
World of computing in 2018



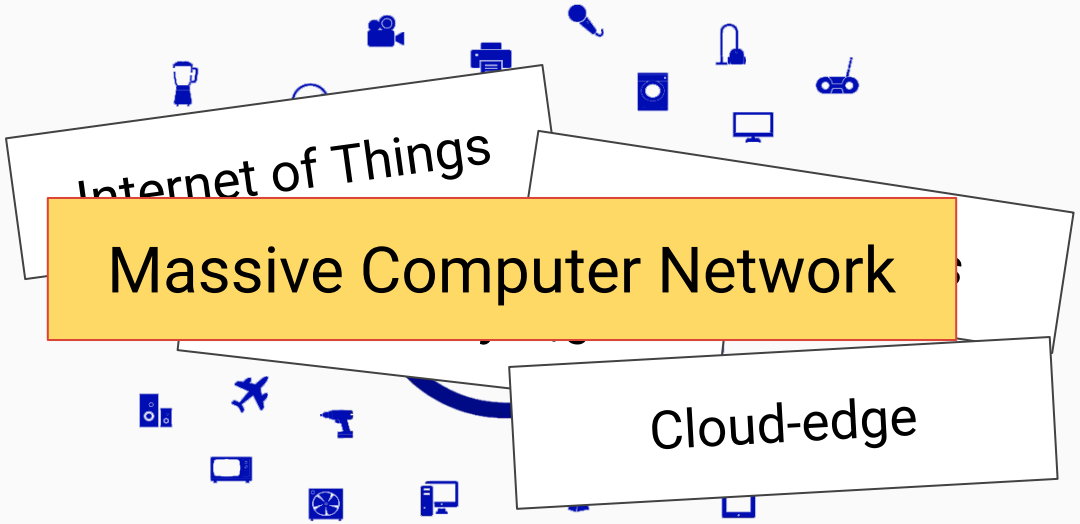
World of computing in 2018



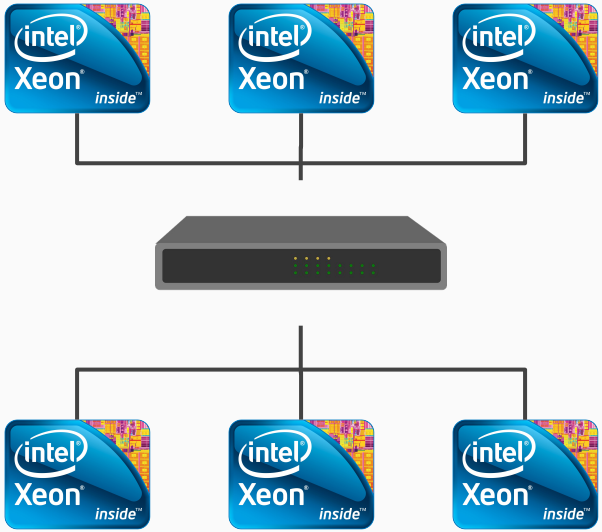
World of computing in 2018



World of computing in 2018

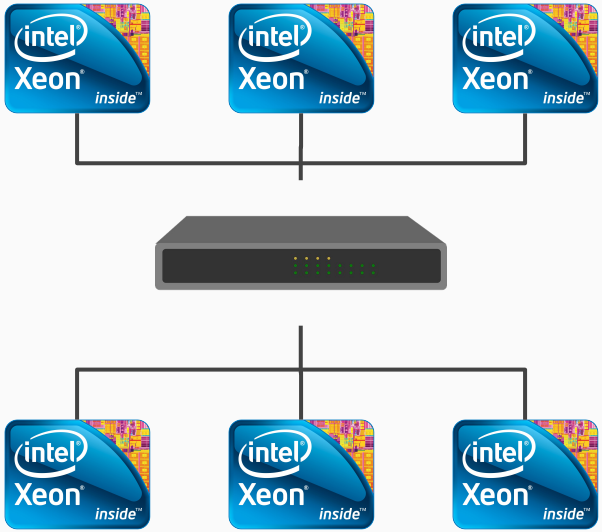


Cloud

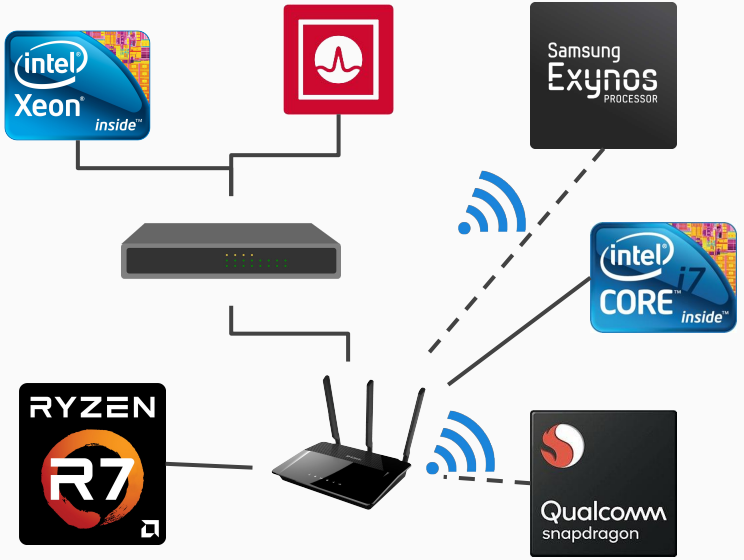

intel
Xeon
inside
intel
Xeon
inside
intel
Xeon
inside
intel
Xeon
inside
intel
Xeon
inside
intel
Xeon
inside

World of computing in 2018

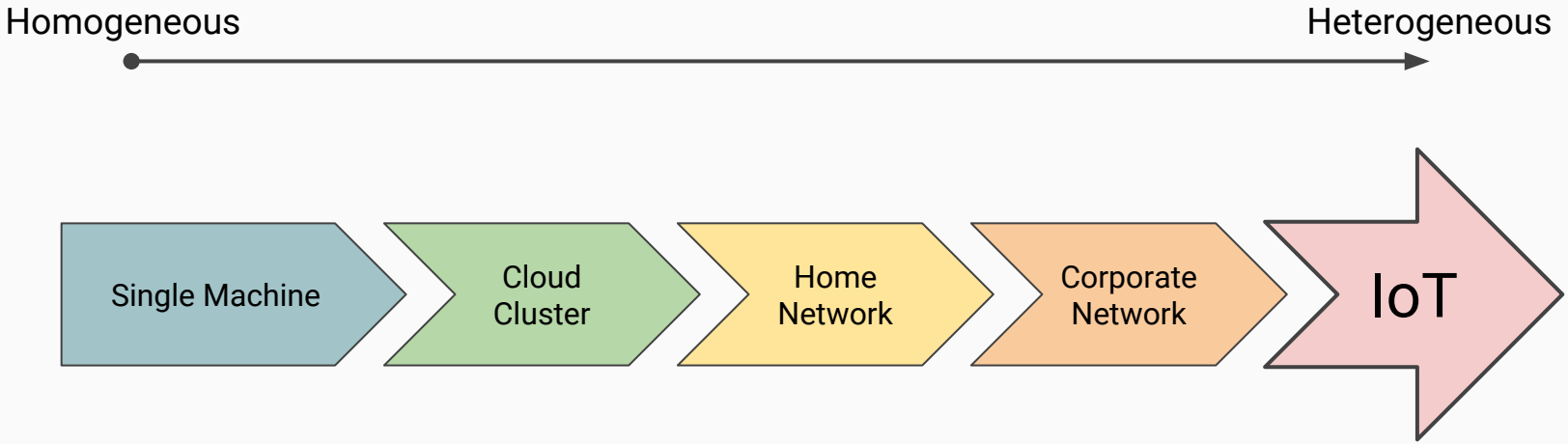
Cloud



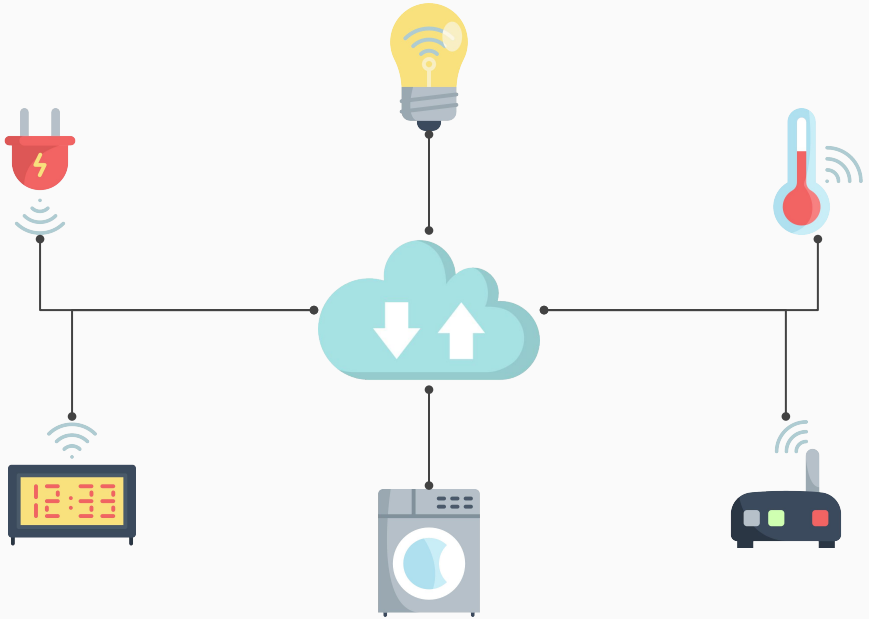
IoT



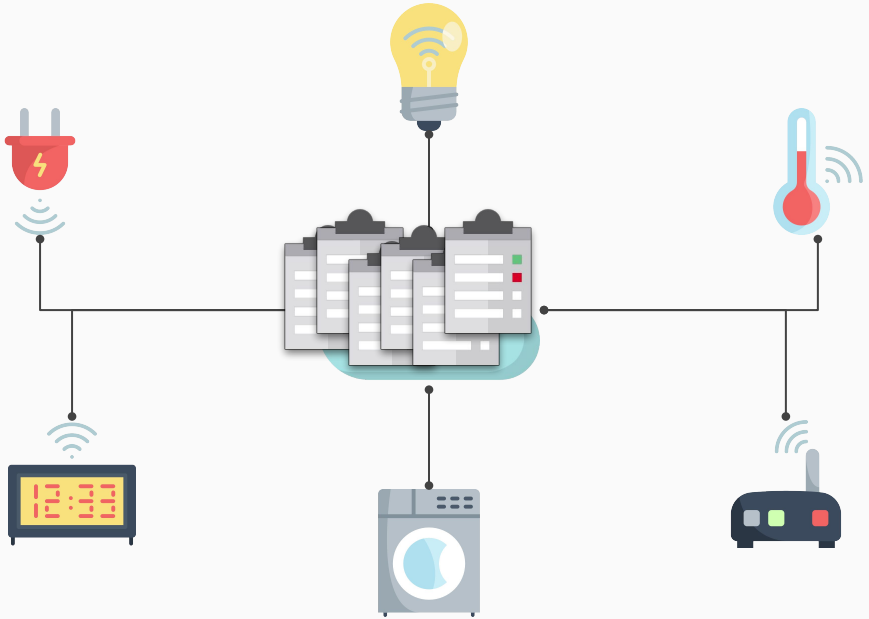
World of computing in 2018



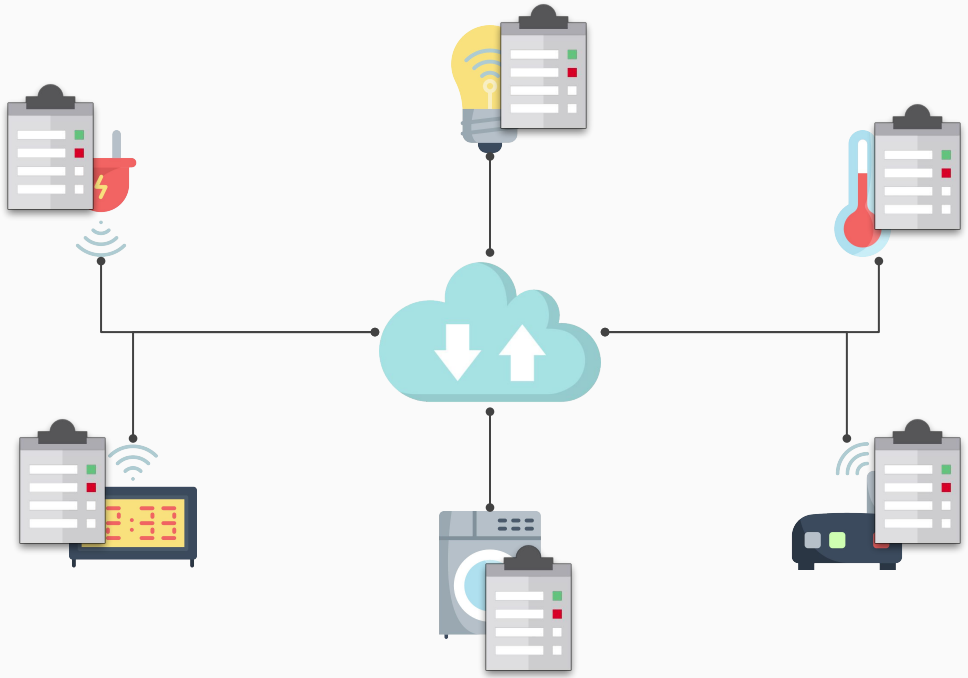
Cloud



Cloud



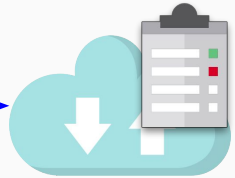
Cloud-Edge

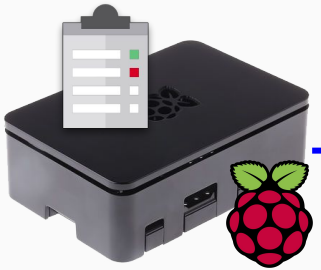


World of computing in 2018

$ 5.00

10 KB / frame
→ 12,960,000 frames / month (5fps)
→ 129.6 GB / month
→ $6.48 / month ($50/TB)
→ **$77.76 / year ~ Raspberry Pi**





$ 70.00

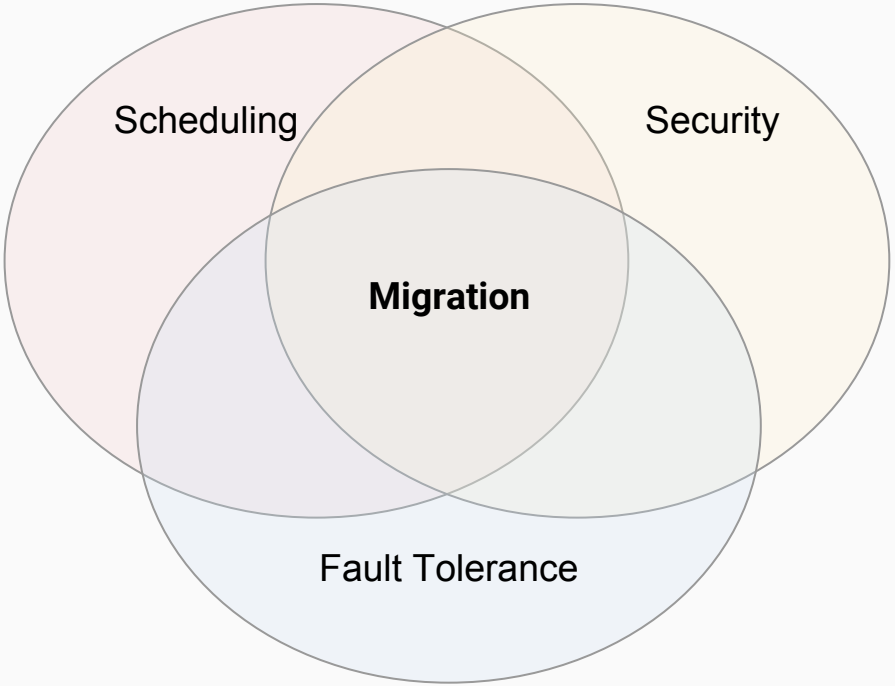
10 MB / request
→ 30 request / month
→ 3.6 GB / year
→ **$2.16 / year**

Motivation

- General Purpose “Edge”
  → Run stateful applications

Motivation

- General Purpose “Edge”
  → Run stateful applications
  → **Need to migrate**

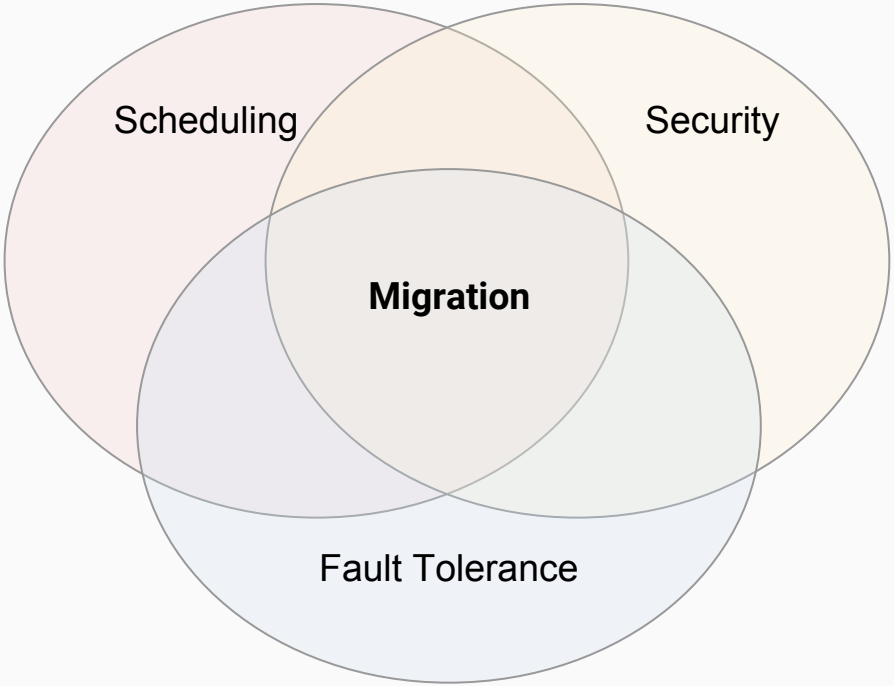


Distributed Computing

Motivation

- General Purpose “Edge”
  → Run stateful applications
  → **Need to migrate**

- Heterogeneous system
- Resource-constrained
  → **Cannot do low-level migration**

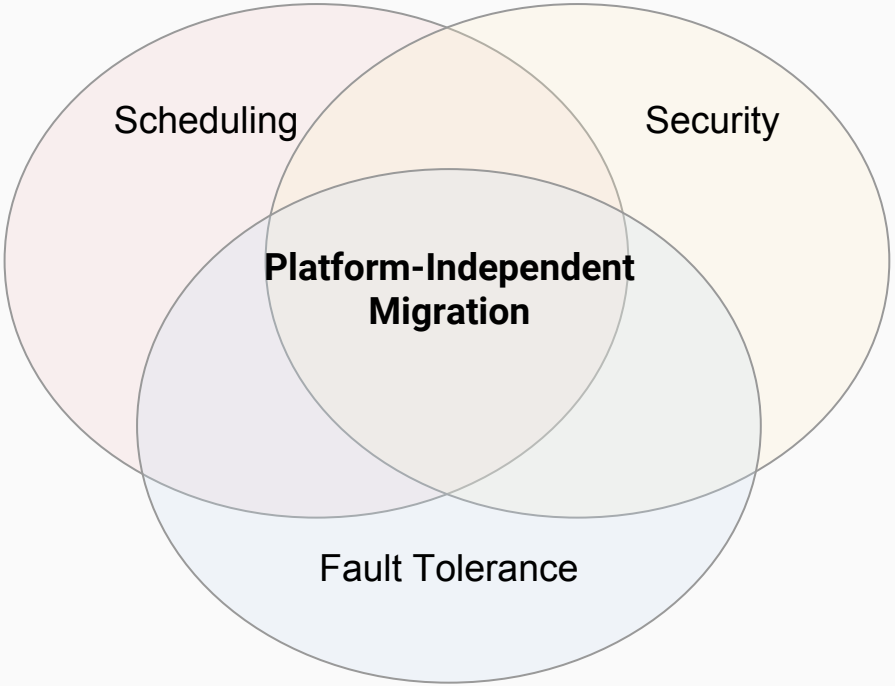


Distributed Computing

Motivation

- General Purpose “Edge”
  → Run stateful applications
  → **Need to migrate**

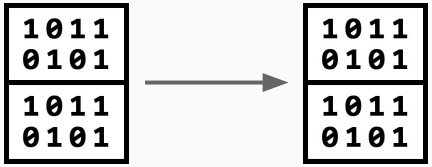
- Heterogeneous system
- Resource-constrained
  → **Cannot do low-level migration**



Distributed Computing in IoT

Low-level Migration

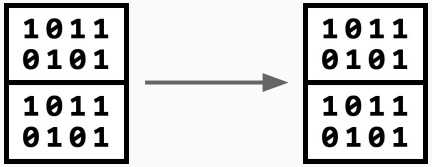
- Program counter
- Registers
- Memory pages



Motivation

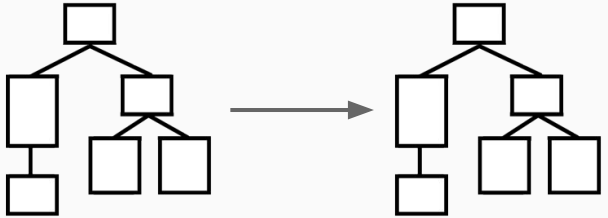
Low-level Migration

- Program counter
- Registers
- Memory pages



High-level Migration

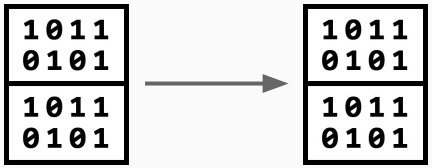
- Stack
- Variables
- Functions



Motivation

Low-level Migration

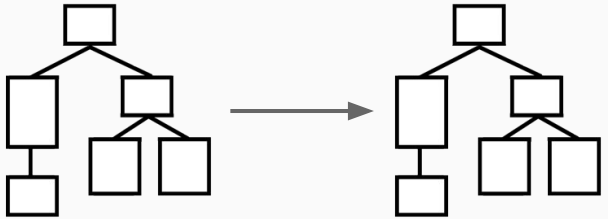
- Program counter
- Registers
- Memory pages



Platform-dependent

High-level Migration

- Stack
- Variables
- Functions



Platform-independent

One language to rule them all

Motivation

One language to rule them all

JavaScript

One language to rule them all

JavaScript

Douglas Crockford

Motivation

One language to rule them all

JavaScript

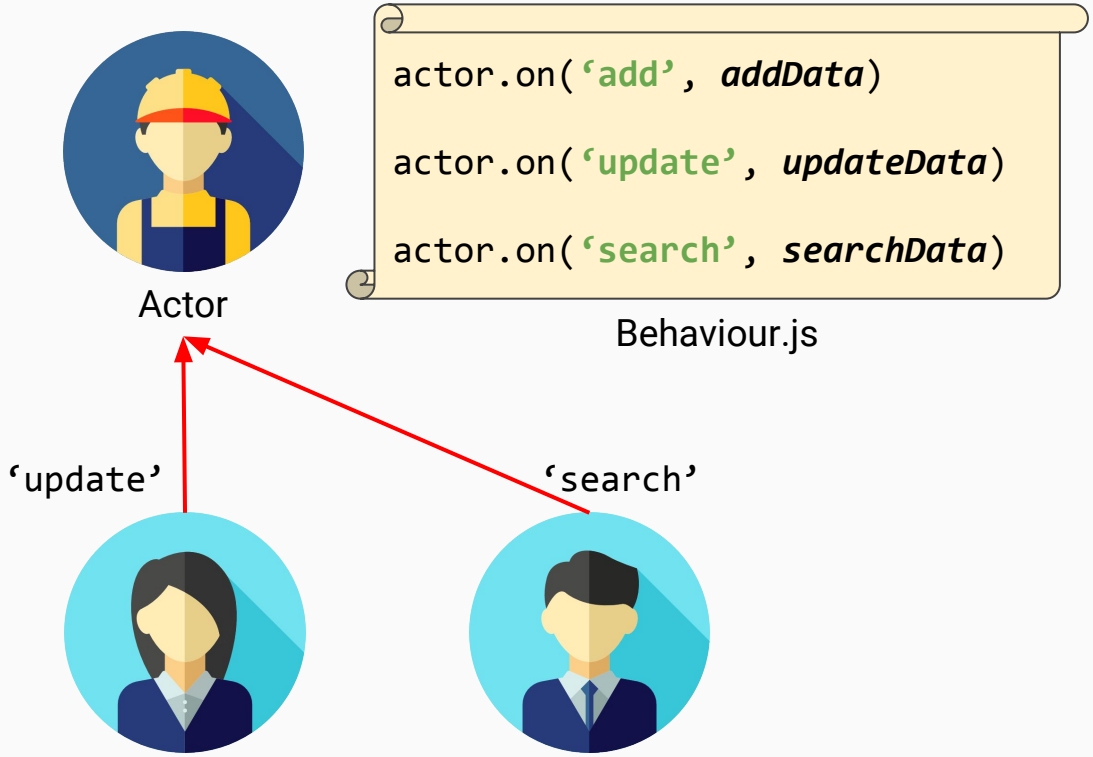
- **Event-driven**
- High-level
- Largest user-base

Motivation

One language to rule them all

JavaScript

- **Event-driven**
- High-level
- Largest user-base

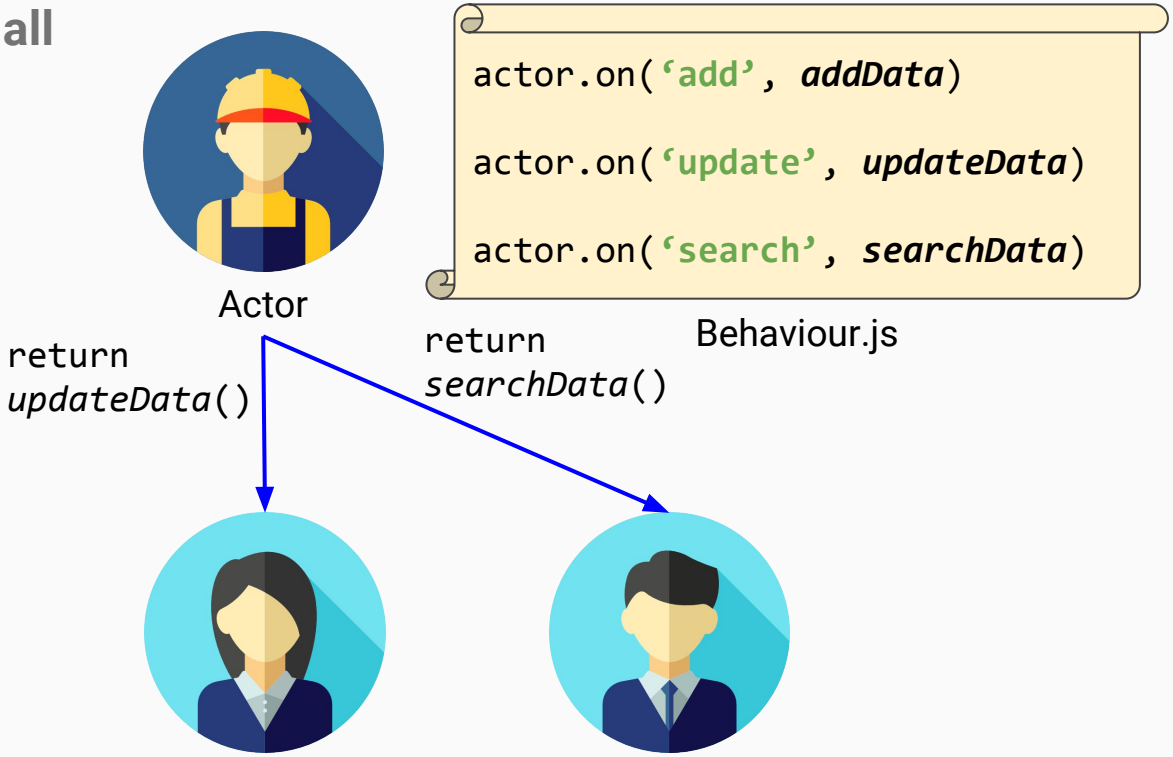


Motivation

One language to rule them all

JavaScript

- **Event-driven**
- High-level
- Largest user-base

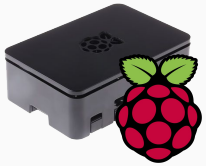


Motivation

One language to rule them all

JavaScript

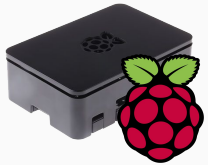
- **Event-driven**
- High-level
- Largest user-base

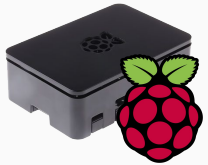


Actor

```
actor.on('add', addData)

actor.on('update', updateData)

actor.on('search', searchData)
```

Behaviour.js

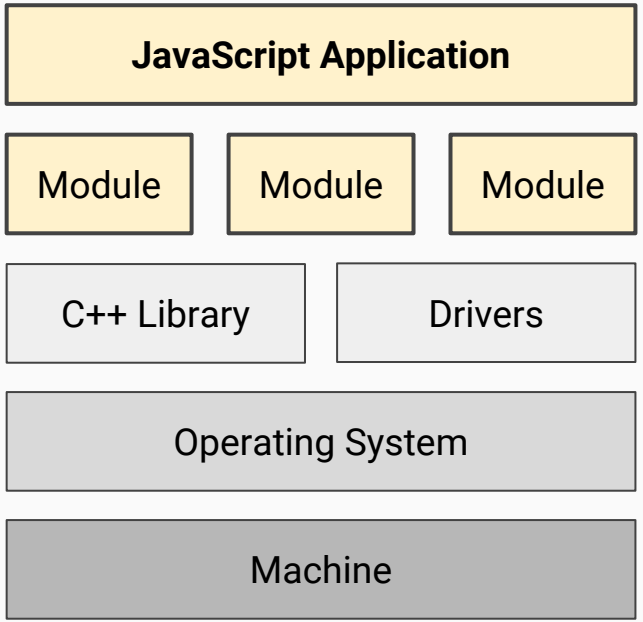




One language to rule them all

JavaScript

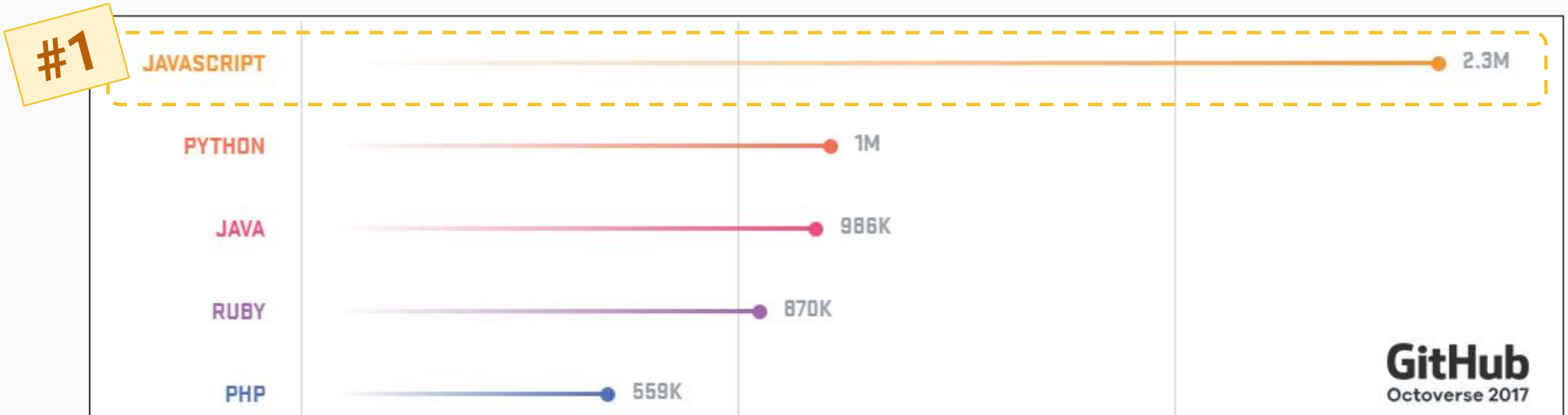
- Event-driven
- **High-level**
- Largest user-base



One language to rule them all

JavaScript

- Event-driven
- High-level
- **Largest user-base**



Related Work

Lo et al.

WWW2013

- Browser to browser

- External Java Server

- One-time migration

Related Work

Lo et al.

WWW2013

- Browser to browser

- External Java Server

- One-time migration

Kwon et al.

WWW2017

- Browser to browser

- Modified VM

Related Work

Lo et al.
WWW2013

- Browser to browser

- External Java Server

- One-time migration

Kwon et al.
WWW2017

- Browser to browser

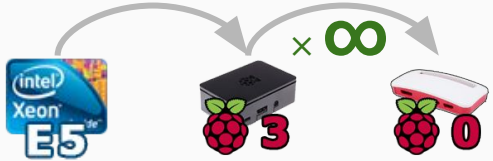
- Modified VM

ThingsMigrate (this)
ECOOP2018

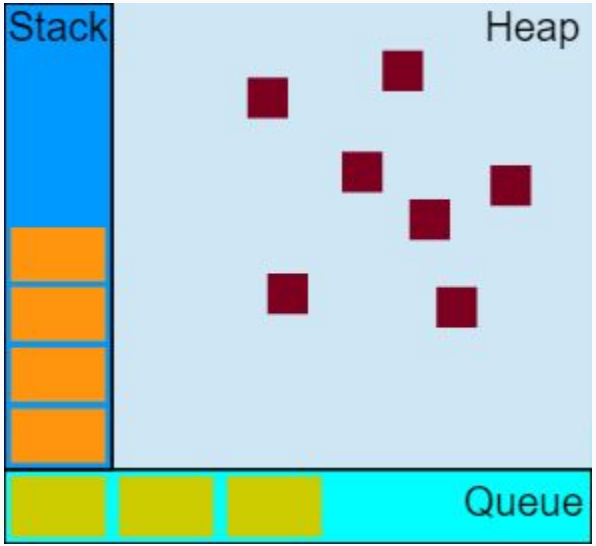
- VM to VM

- Only JavaScript

- Multi-hop migration



JavaScript process



source: developer.mozilla.org

JavaScript process

source: developer.mozilla.org

Main Challenges

1. **Closures**
2. Events

```
function PiggyBank(){

   var balance = 0
   var deposit = function(amount){

      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){

      bank(100)
}, 1000)
```

Main Challenges

1. Closures
2. **Events**

```
function PiggyBank(){

   var balance = 0
   var deposit = function(amount){

      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){

      bank(100)
}, 1000)
```

Approach

Given a JavaScript program:

1. *Instrumentation* - **Modify code** so we can capture state

Approach

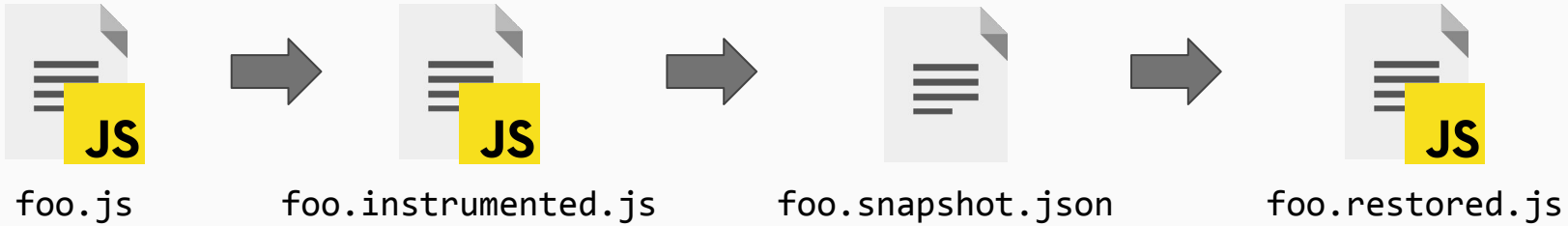
Given a JavaScript program:

1. *Instrumentation* - **Modify code** so we can capture state
2. *Serialization* - Serialize state into a **snapshot**

Approach

Given a JavaScript program:

1. *Instrumentation* - **Modify code** so we can capture state
2. *Serialization* - Serialize state into a **snapshot**
3. *Restoration* - **Generate code** from snapshot

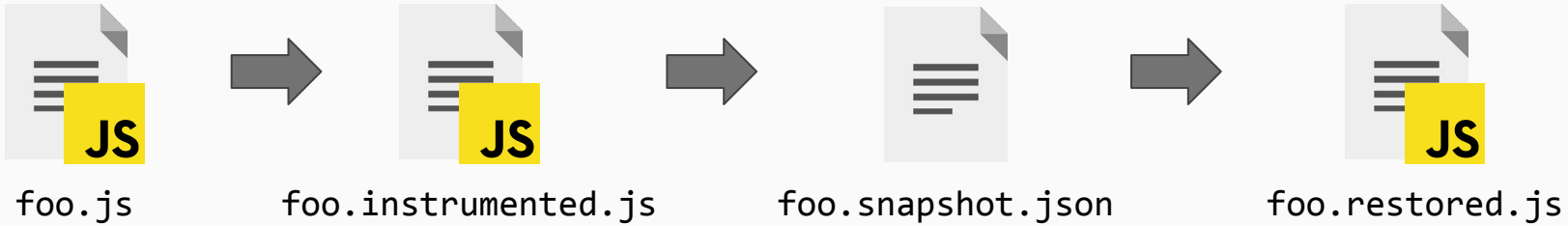


Approach

Given a JavaScript program:

1. *Instrumentation* - **Modify code** so we can capture state
2. *Serialization* - Serialize state into a **snapshot**
3. *Restoration* - **Generate code** from snapshot

```
program = restore(snapshot(program))
```



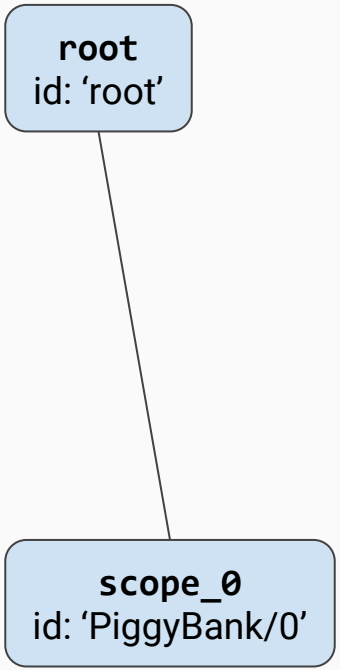
Implementation - Instrumentation


```
function PiggyBank(){

   var balance = 0
   var deposit = function(amount){

      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){

      bank(100)
}, 1000)
```

root
id: ‘root’

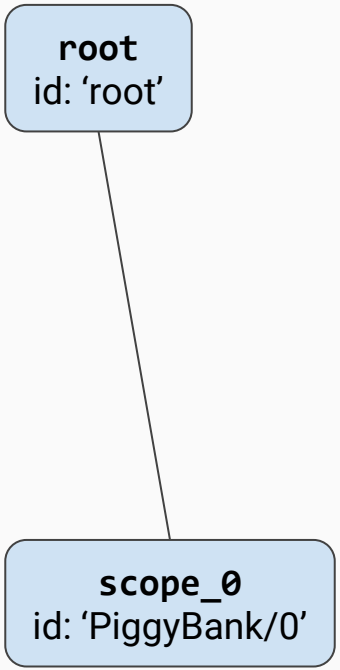
```
var root = new Scope()
function PiggyBank(){

   var balance = 0
   var deposit = function(amount){

      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){

      bank(100)
}, 1000)
```

Implementation - Instrumentation



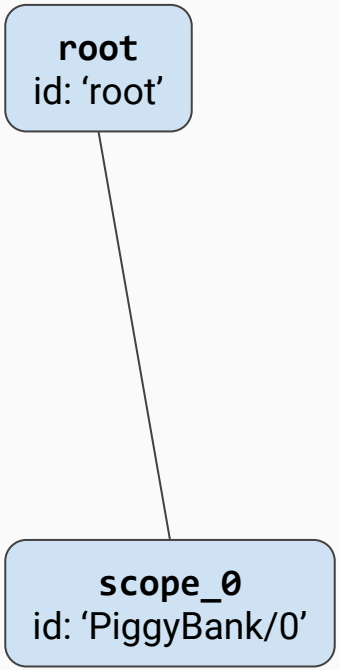

```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   var deposit = function(amount){

      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){

      bank(100)
}, 1000)
```

Implementation - Instrumentation



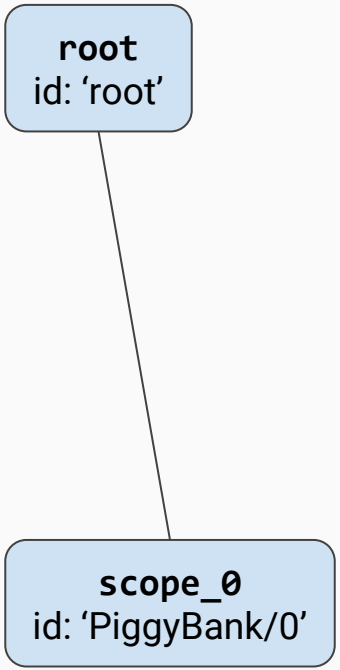

```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){

      bank(100)
}, 1000)
```

Implementation - Instrumentation



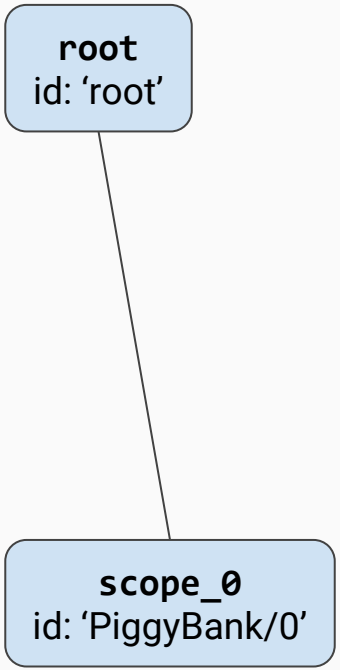

```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}, 1000)
```

Implementation - Instrumentation



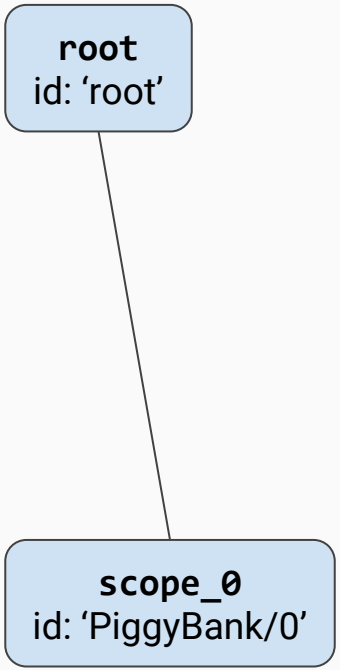

```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount
   }
   return deposit
}
var bank = PiggyBank()
setInterval(
   function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}, 1000)
```

Implementation - Instrumentation



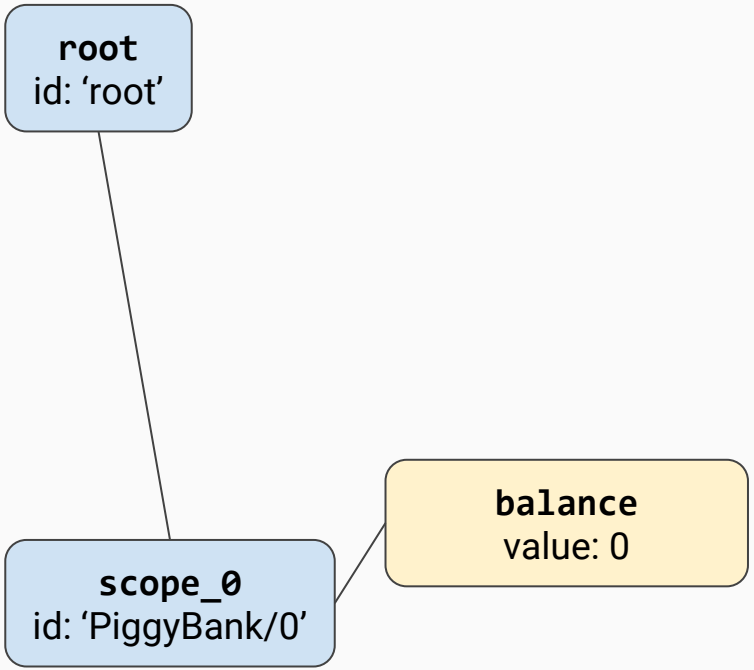

```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount
   }
   return deposit
}

/* truncated */
```

Implementation - Instrumentation

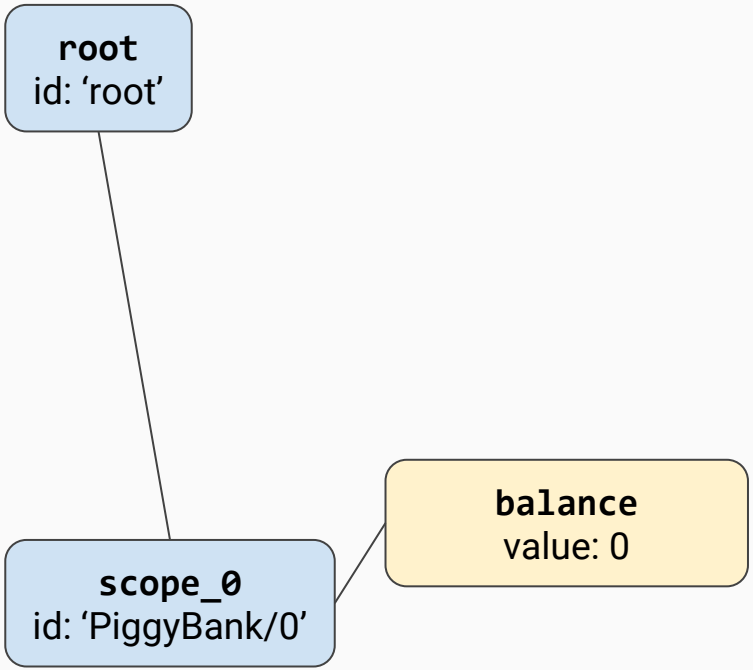



```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0

   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount

   }

   return deposit
}

/* truncated */
```

Implementation - Instrumentation

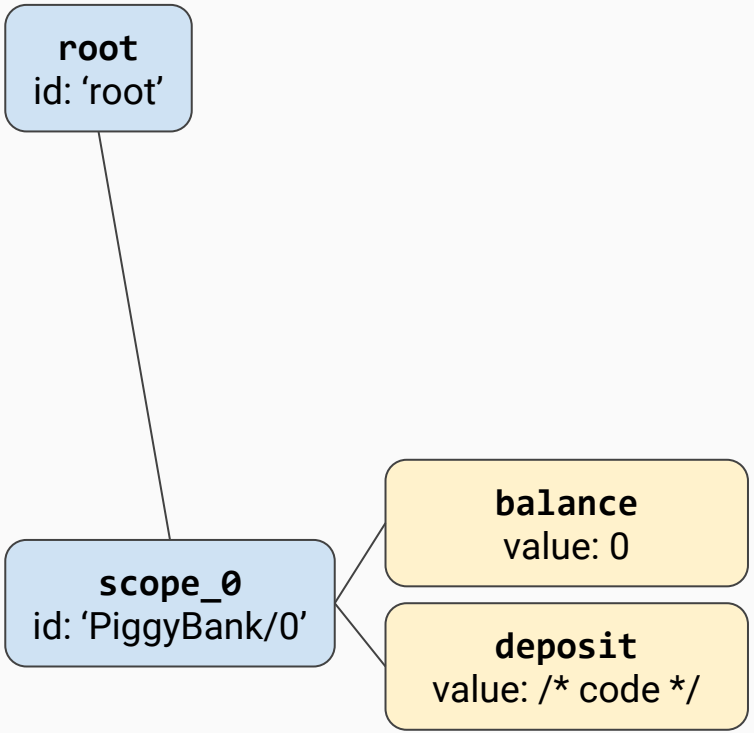



```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   scope_0.vars.balance = balance
   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount

   }

   return deposit
}

/* truncated */
```



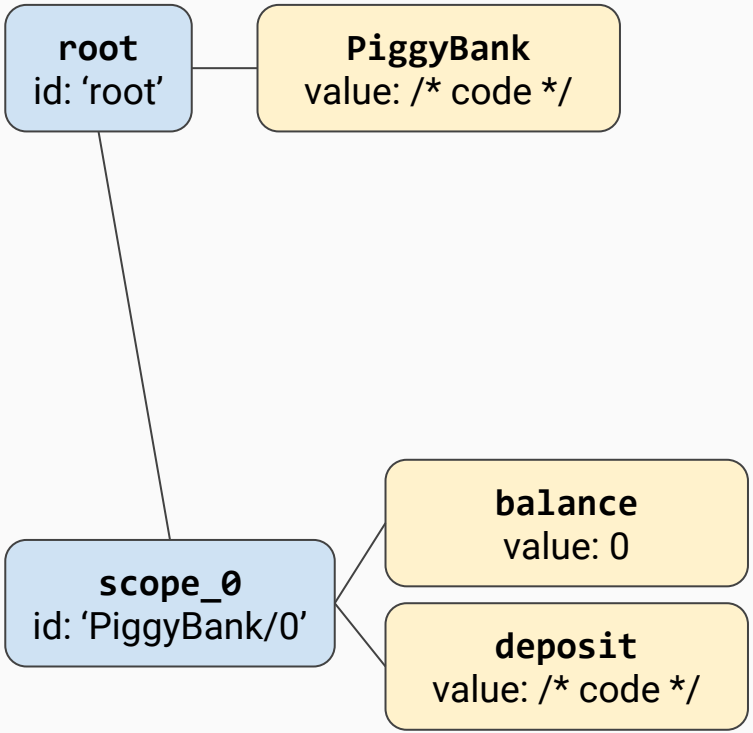

```
var root = new Scope()
function PiggyBank(){
   var scope_0 = new Scope(root)
   var balance = 0
   scope_0.vars.balance = balance
   var deposit = function(amount){
      var scope_0_0 = new Scope(scope_0)
      balance += amount
      scope_0.vars.balance = balance
   }

   return deposit
}

/* truncated */
```

Implementation - Instrumentation



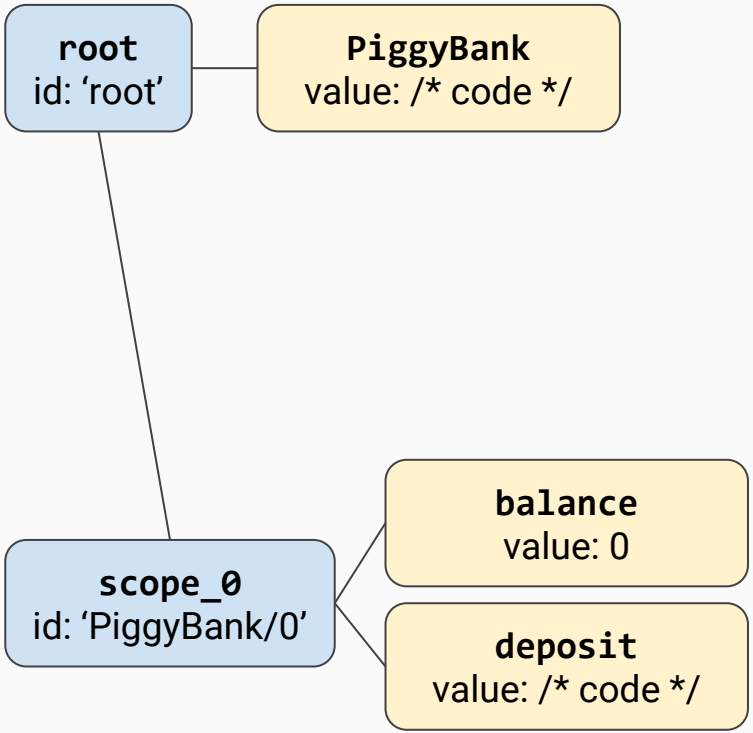

```
var root = new Scope()
function PiggyBank(){
    var scope_0 = new Scope(root)
    var balance = 0
    scope_0.vars.balance = balance
    var deposit = function(amount){
        var scope_0_0 = new Scope(scope_0)
        balance += amount
        scope_0.vars.balance = balance
    }
    scope_0.addFunction(deposit)
    return deposit
}

/* truncated */
```

Implementation - Instrumentation



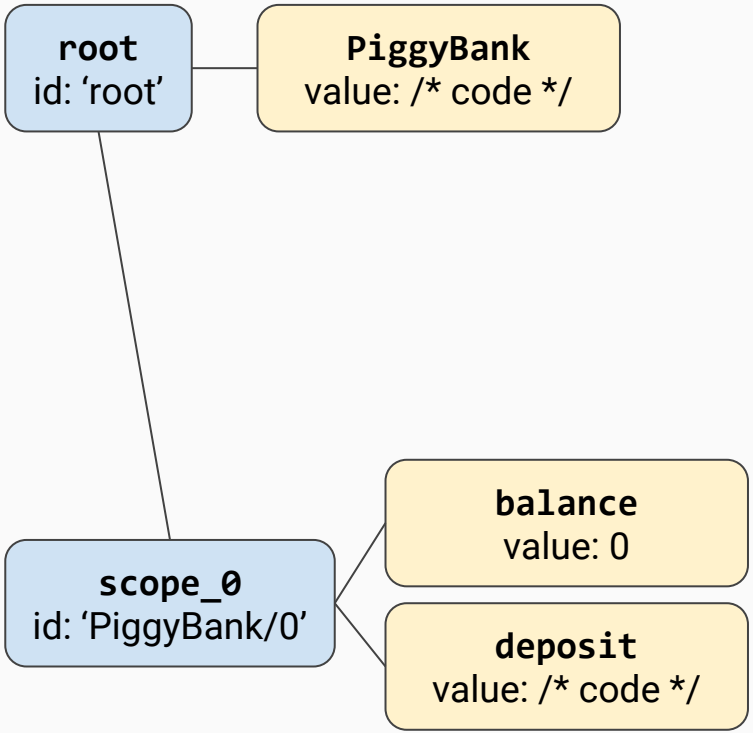

```
var root = new Scope()
function PiggyBank(){
    var scope_0 = new Scope(root)
    var balance = 0
    scope_0.vars.balance = balance
    var deposit = function(amount){
        var scope_0_0 = new Scope(scope_0)
        balance += amount
        scope_0.vars.balance = balance
    }
    scope_0.addFunction(deposit)
    return deposit
}
root.addFunction(PiggyBank)
/* truncated */
```

Implementation - Instrumentation



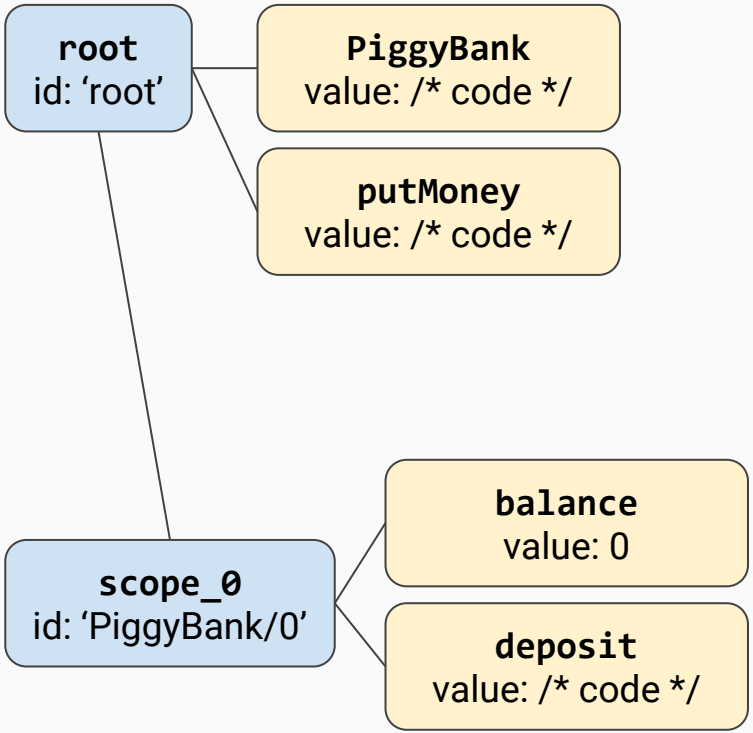

```
var root = new Scope()
function PiggyBank(){
    /* truncated */
}
root.addFunction(PiggyBank)
/* truncated */
```

Implementation - Instrumentation

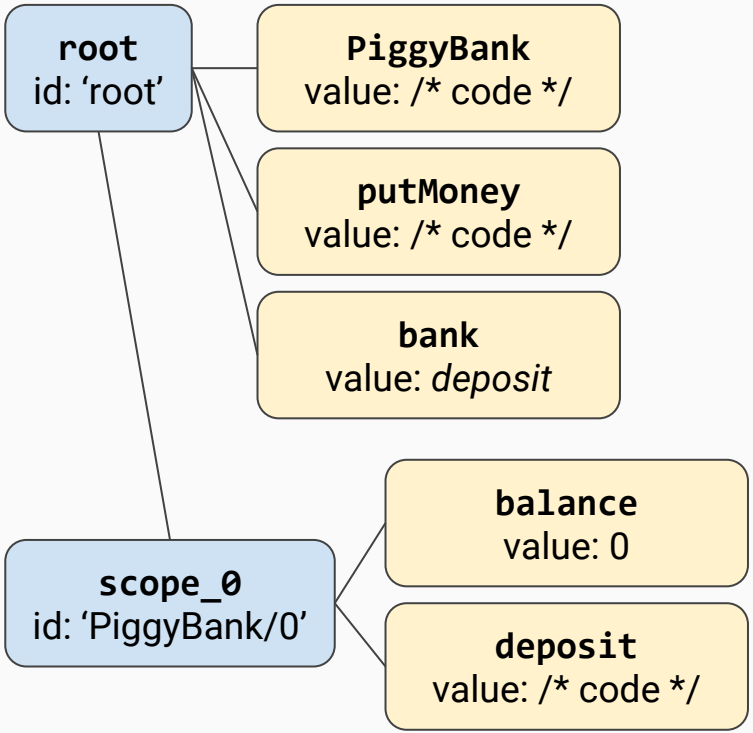



```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()

setInterval(
   function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}, 1000)
```

Implementation - Instrumentation

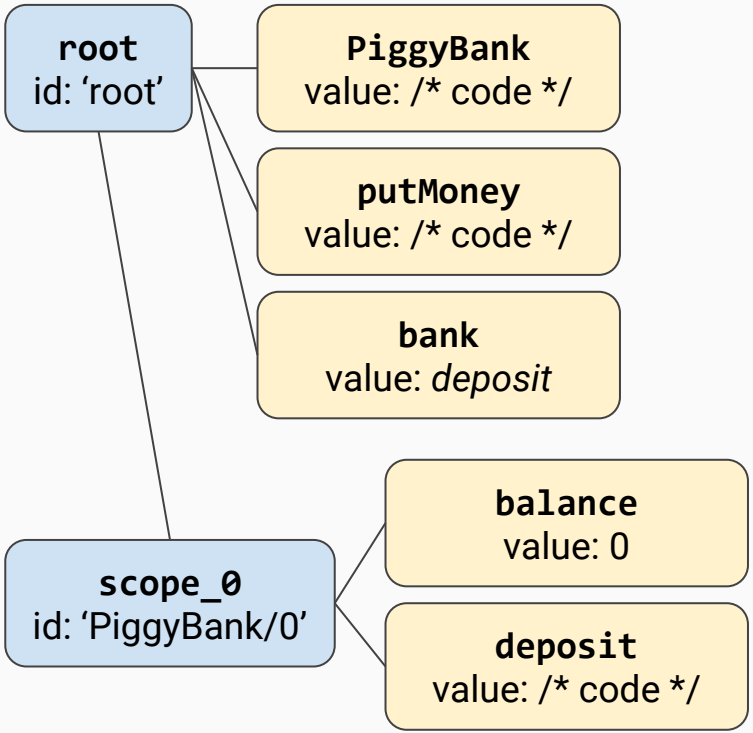



```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()

setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```



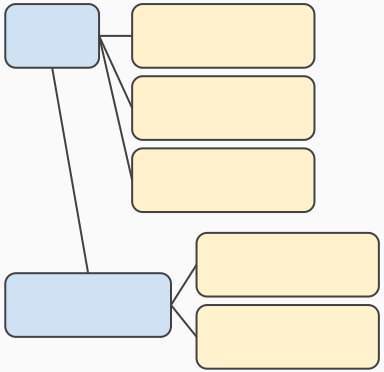

```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()
root.vars.bank = bank
setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```

Implementation - Instrumentation



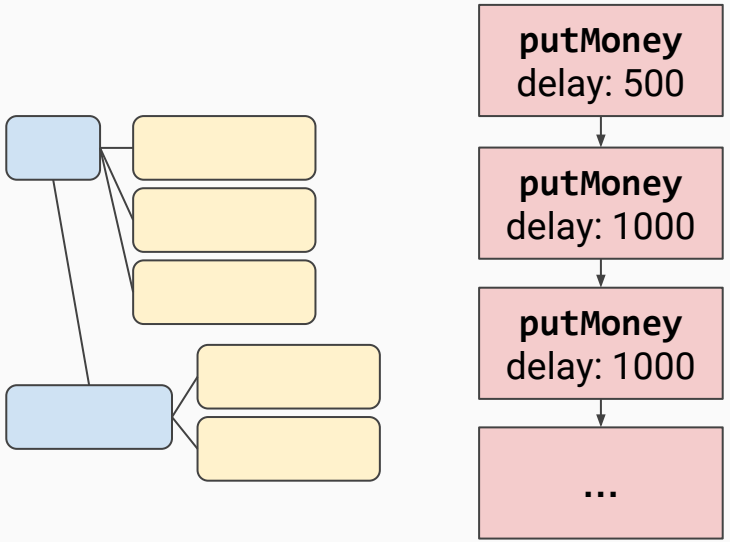

```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()
root.vars.bank = bank
setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```

Implementation - Instrumentation



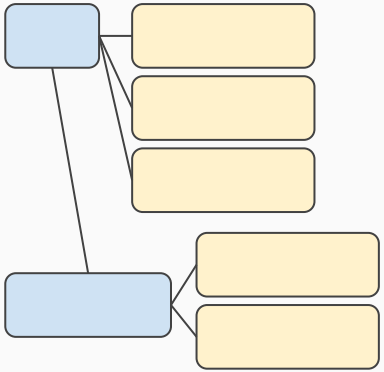
```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()
root.vars.bank = bank
setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```

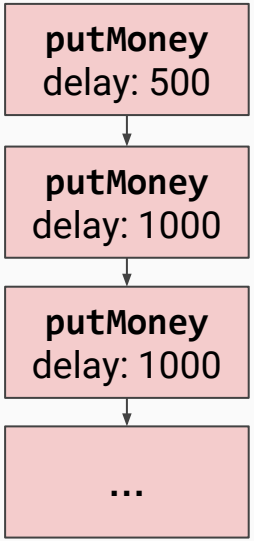
Implementation - Instrumentation




```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()
root.vars.bank = bank
things.setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```

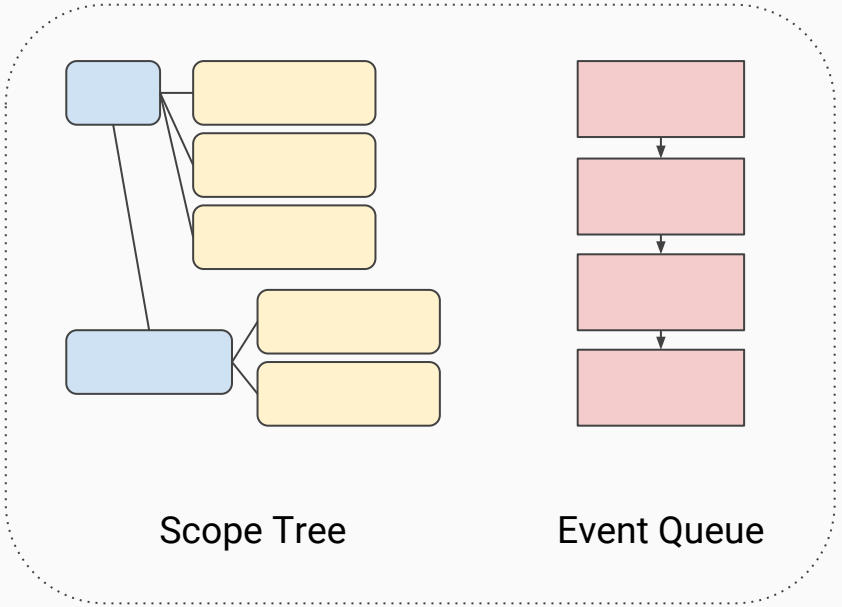
Implementation - Instrumentation





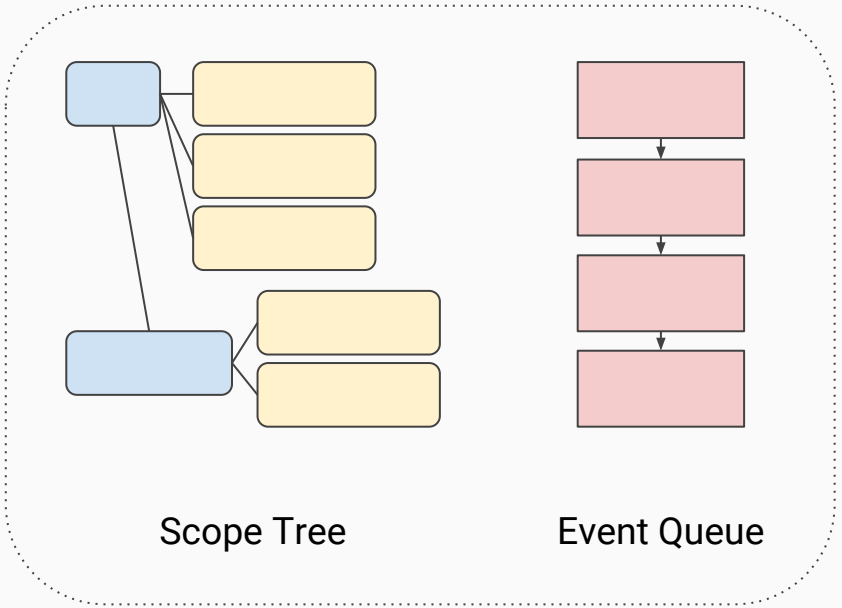

```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()
root.vars.bank = bank
things.setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```

Implementation - Instrumentation



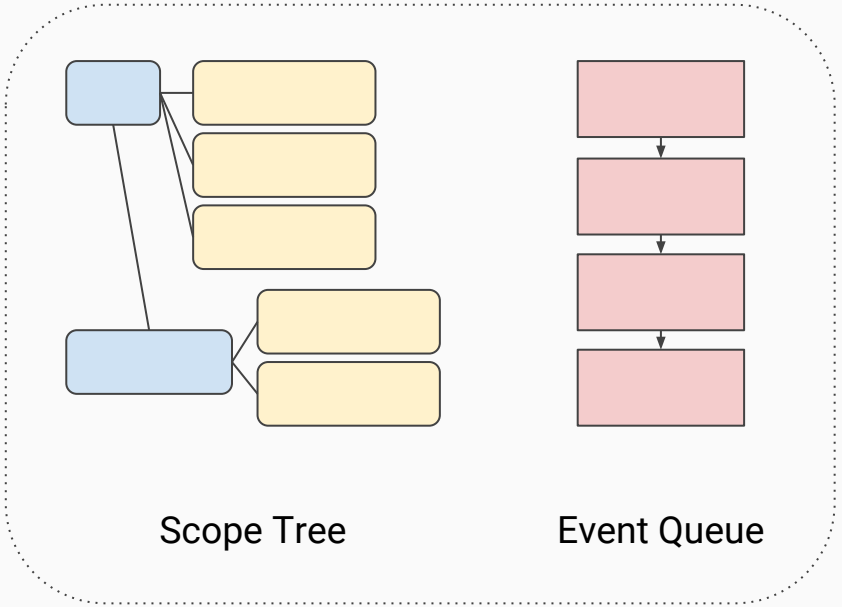

```
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
root.addFunction(PiggyBank)
var bank = PiggyBank()
root.vars.bank = bank
things.setInterval(
   root.addFunction(function putMoney(){
      var scope_1 = new Scope(root)
      bank(100)
}), 1000)
```

Implementation - Instrumentation



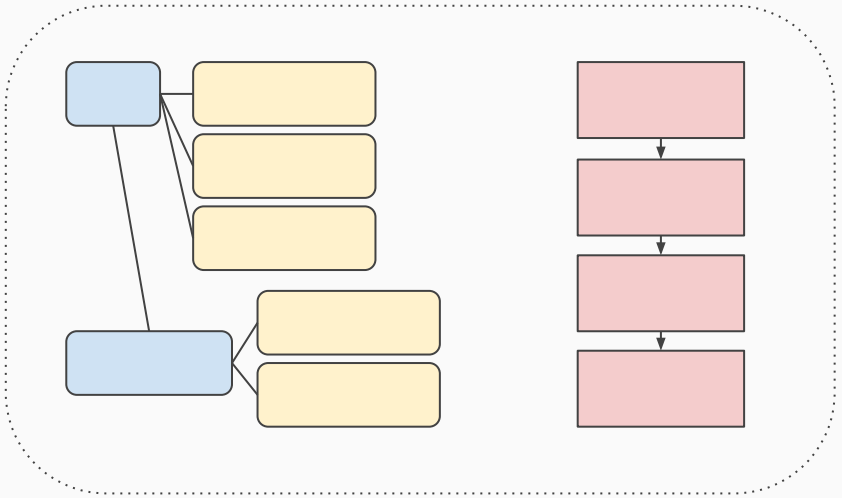

```
var root = new Scope()
function PiggyBank(){
    /* truncated */
}
/* truncated */
```

Implementation - Instrumentation




```
var root = new Scope()
function PiggyBank(){
    /* truncated */
}
/* truncated */
```

Implementation - Instrumentation

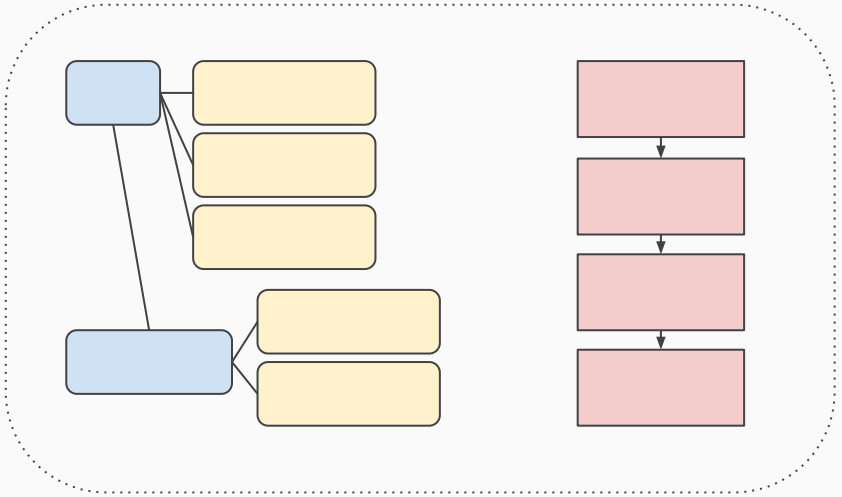


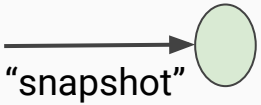
Pubsub

```
var pubsub = new Pubsub('mqtt://1.2.3.4')
```

```
var root = new Scope()
function PiggyBank(){
    /* truncated */
}
/* truncated */
```

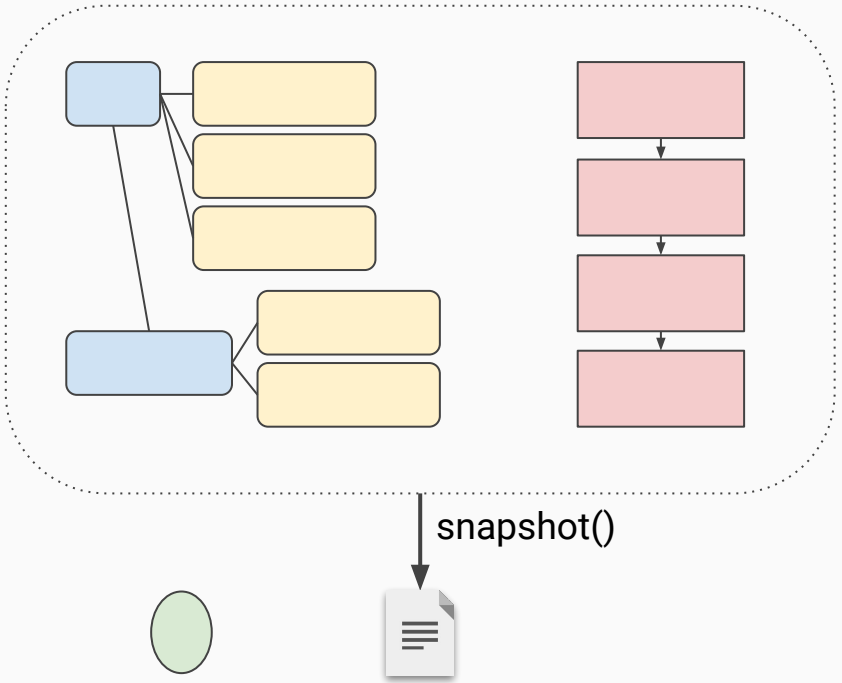
Implementation - Instrumentation





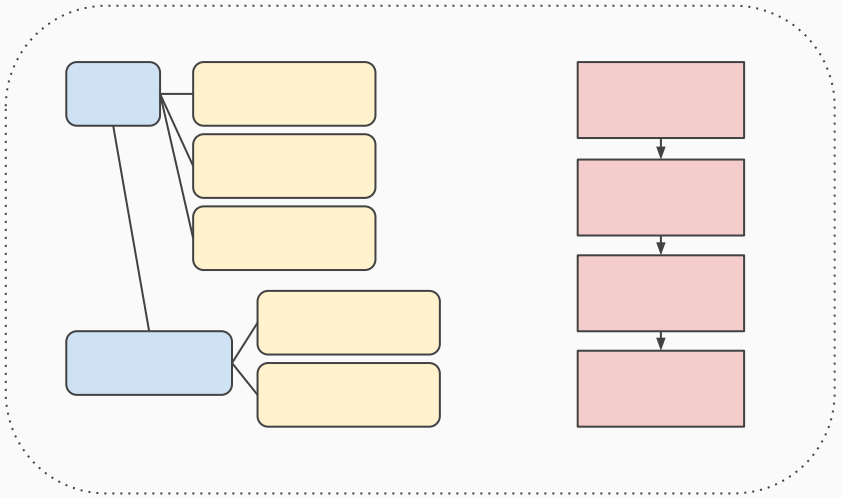

```
var pubsub = new Pubsub(‘mqtt://1.2.3.4’)
pubsub.subscribe(‘snapshot’, function(){



})
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
/* truncated */
```

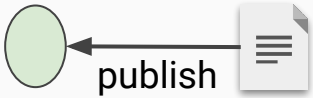
Implementation - Instrumentation




```
var pubsub = new Pubsub('mqtt://1.2.3.4')
pubsub.subscribe('snapshot', function(){
    var snapshot = root.snapshot()

})
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
/* truncated */
```

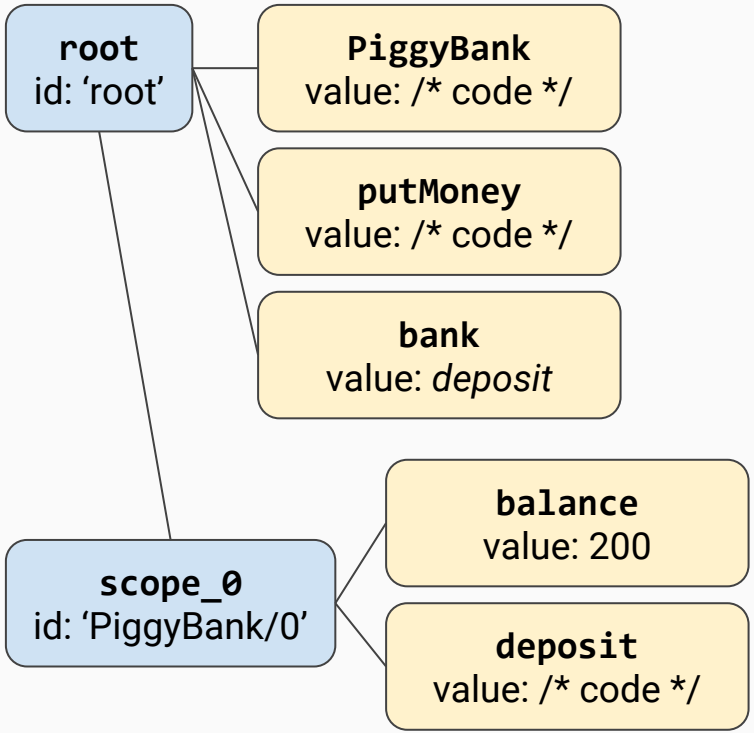
Implementation - Instrumentation



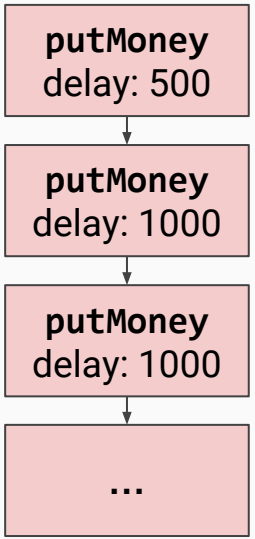



```
var pubsub = new Pubsub('mqtt://1.2.3.4')
pubsub.subscribe('snapshot', function(){
    var snapshot = root.snapshot()
    pubsub.publish('snapshots', snapshot)
})
var root = new Scope()
function PiggyBank(){
   /* truncated */
}
/* truncated */
```

Implementation - Serialization

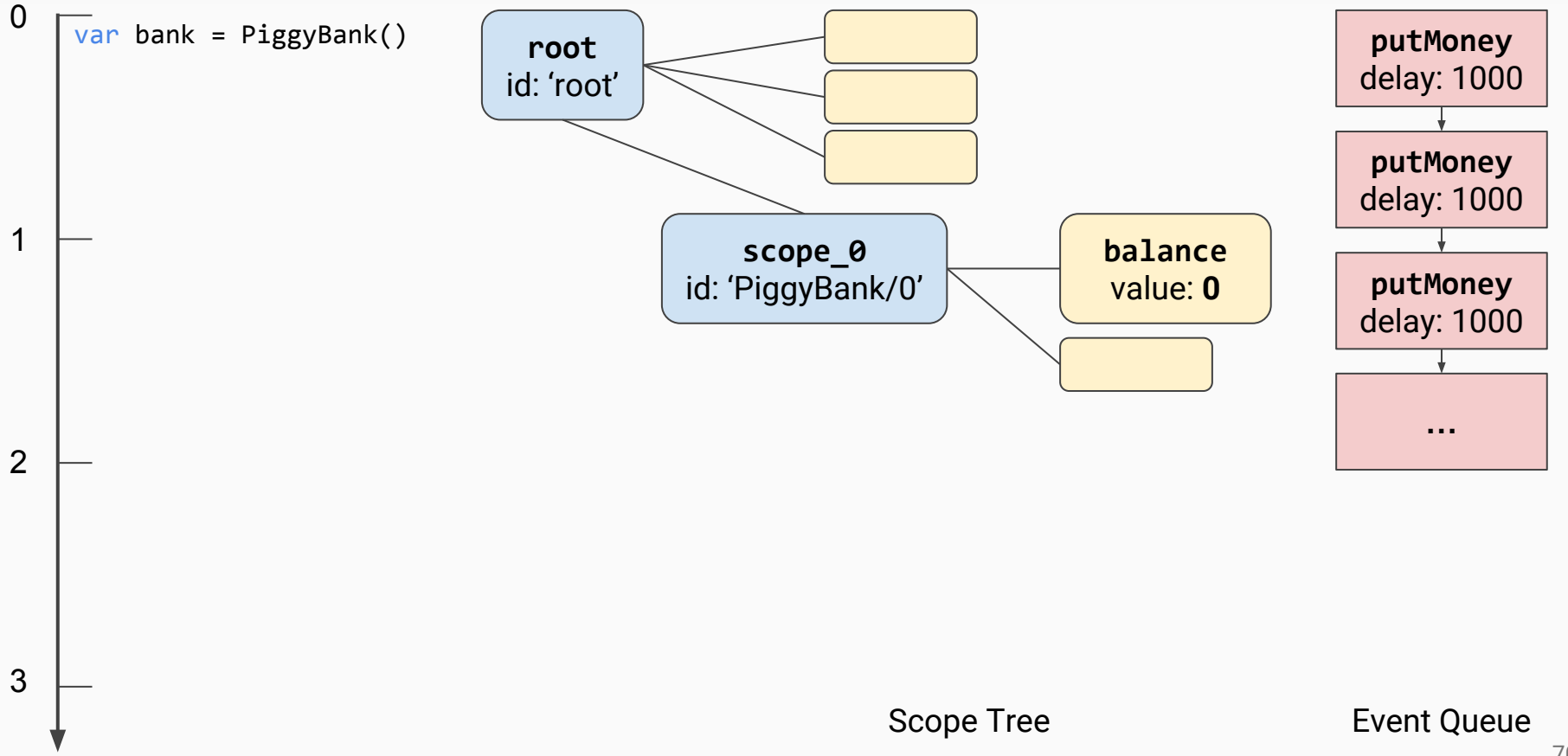


Scope Tree

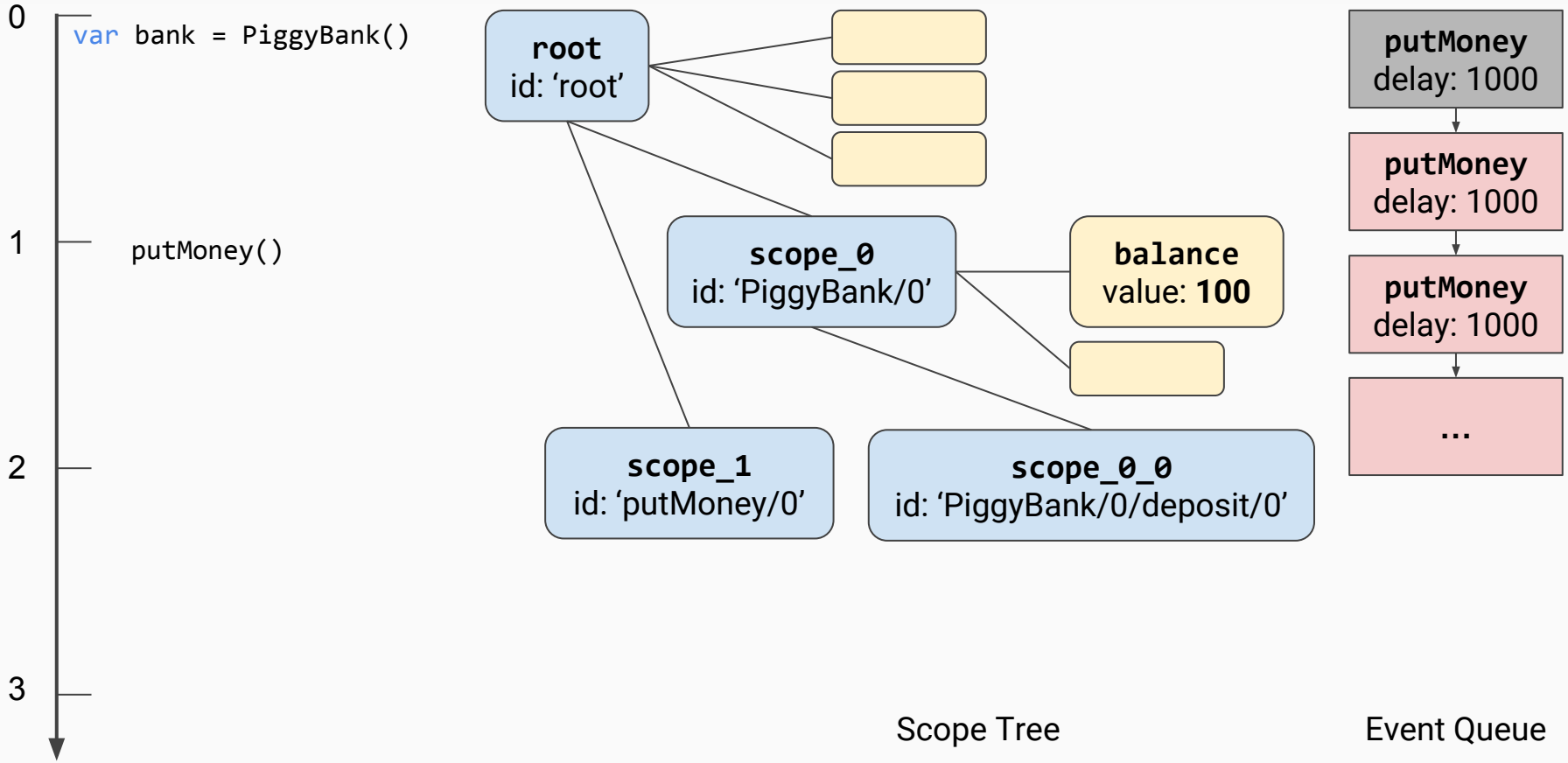


Event Queue

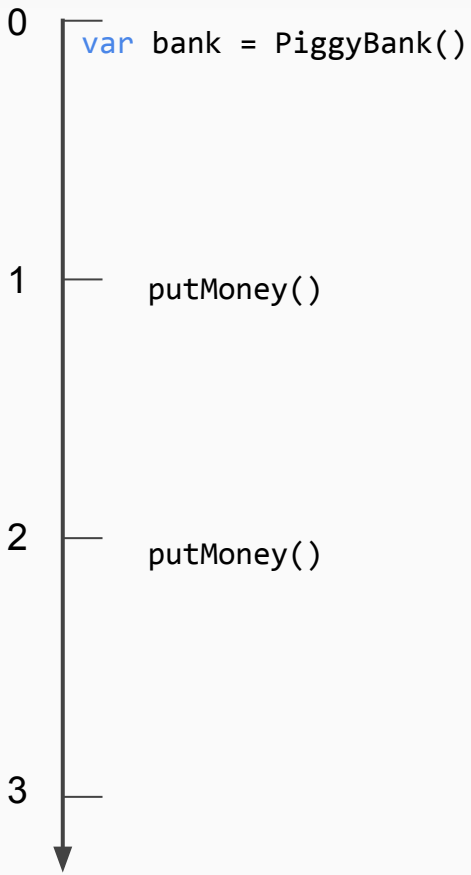
Implementation - Serialization

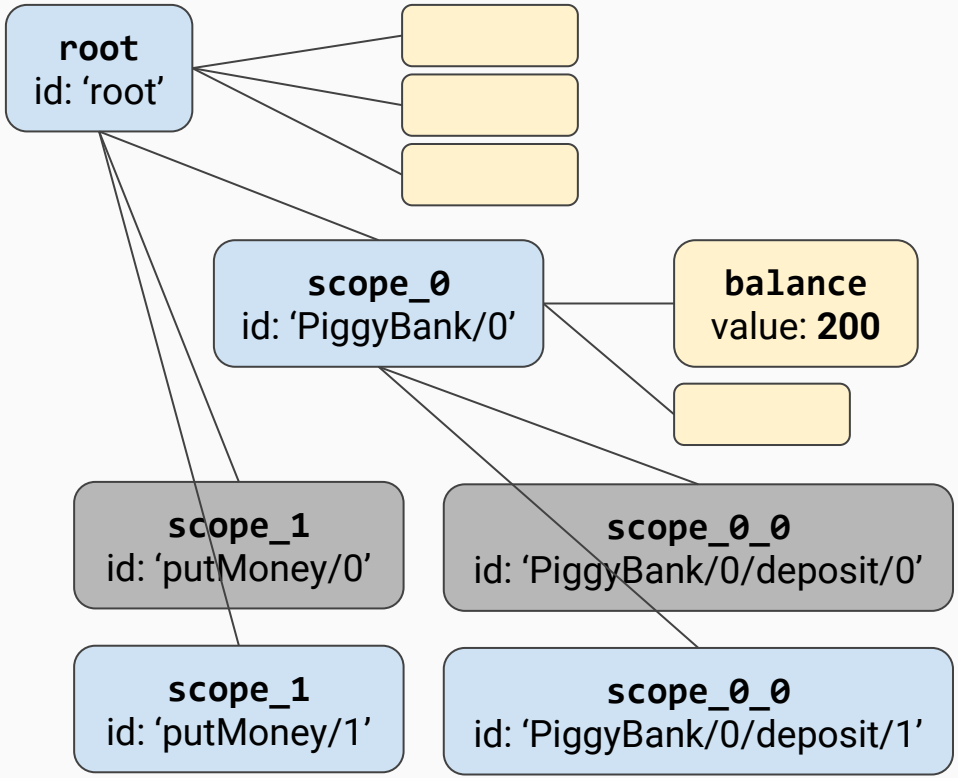


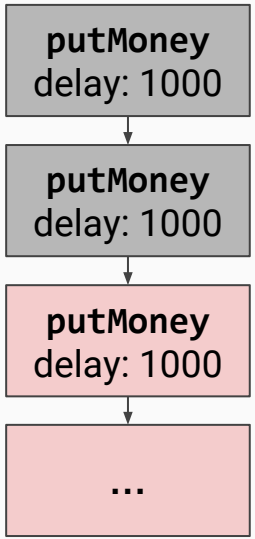
Implementation - Serialization



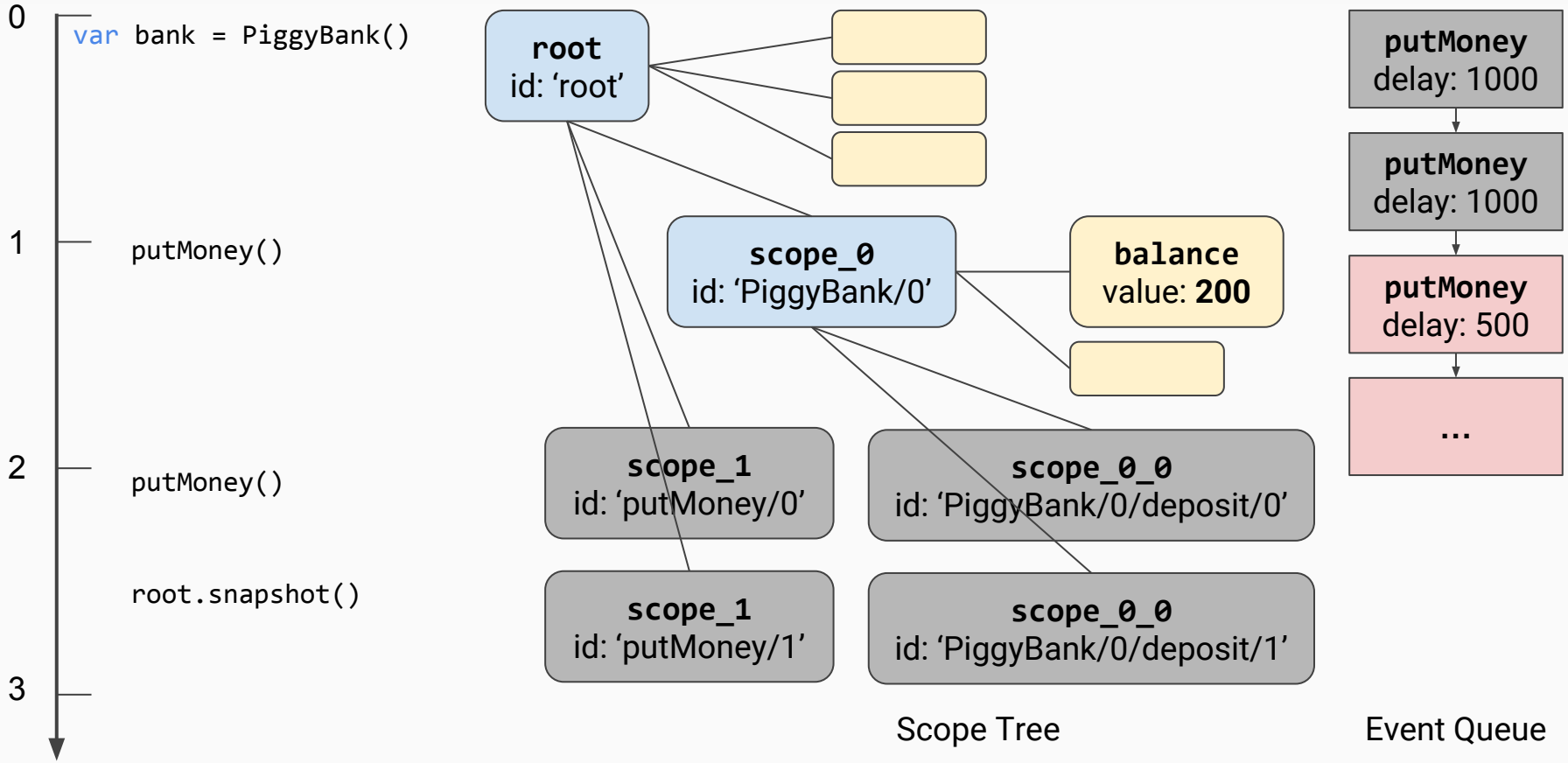
Implementation - Serialization



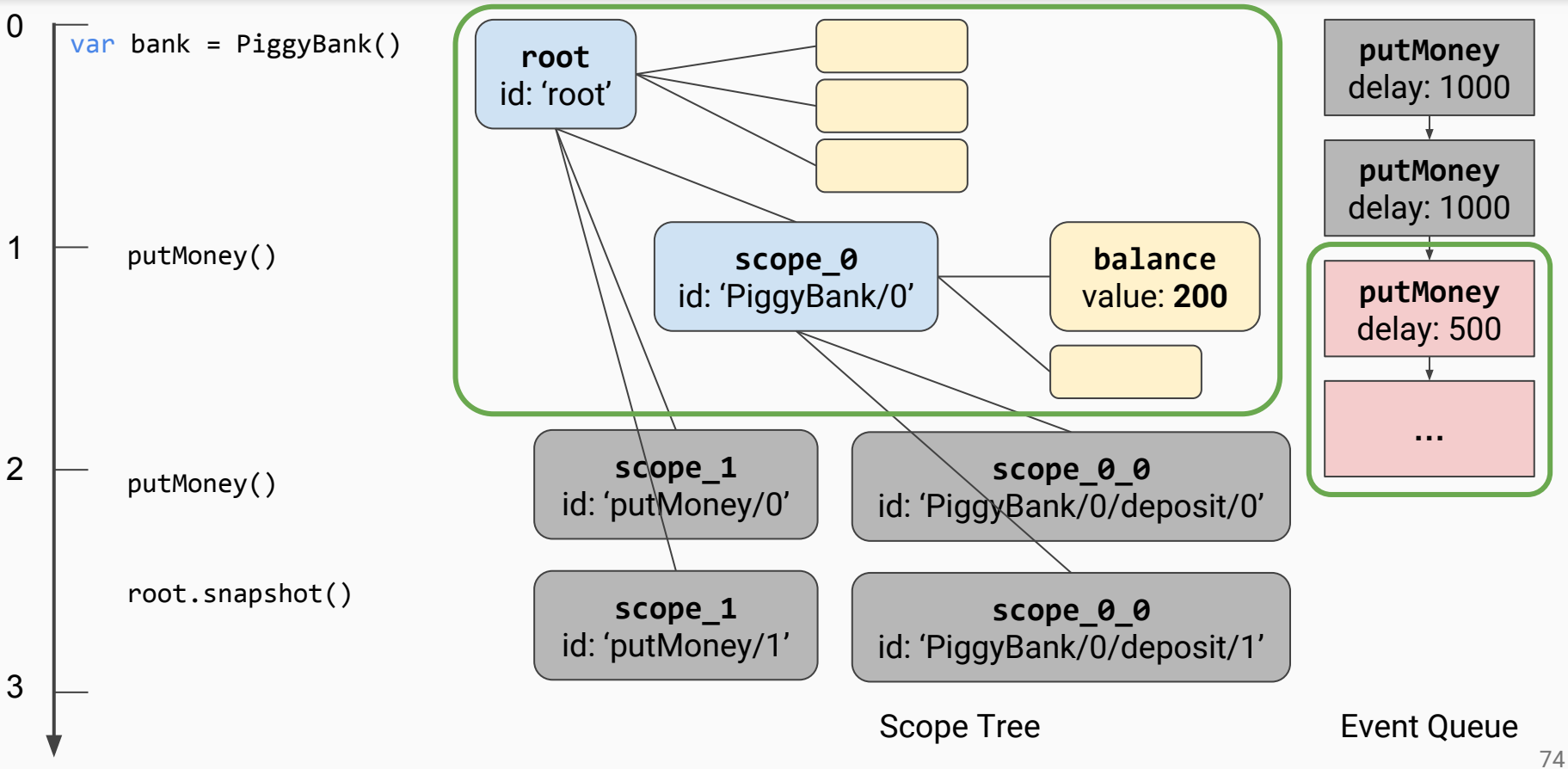




Implementation - Serialization

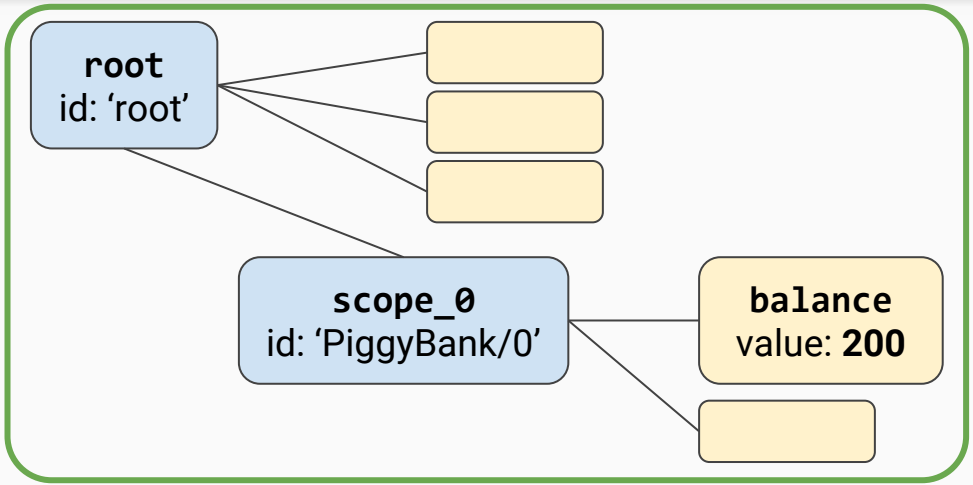


Implementation - Serialization

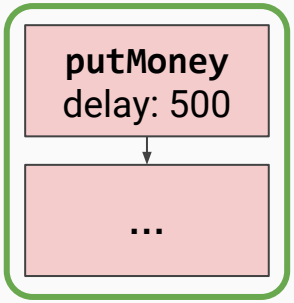


Implementation - Serialization


```
0  var bank = PiggyBank()
1  putMoney()
2  putMoney()
   root.snapshot()
3
```



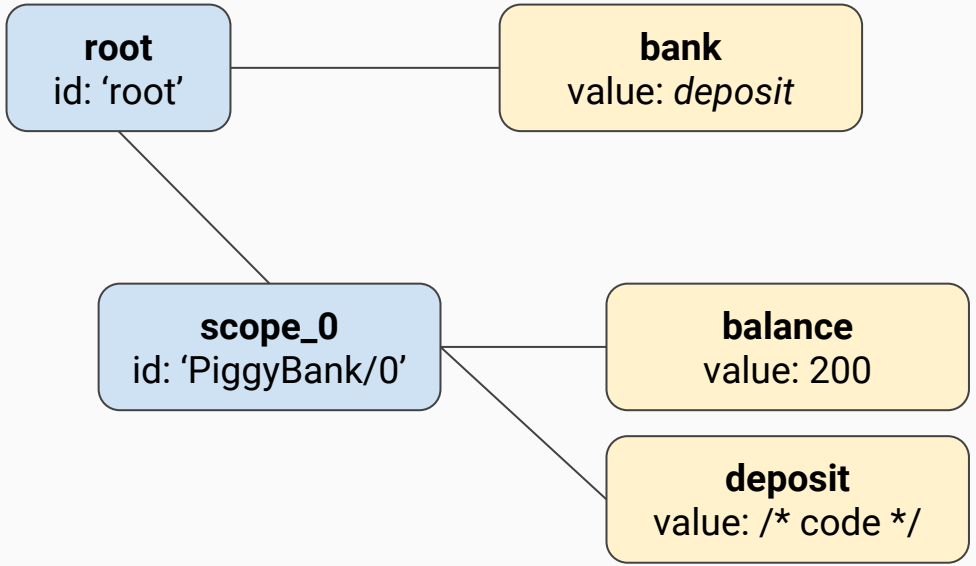
Scope Tree



Event Queue

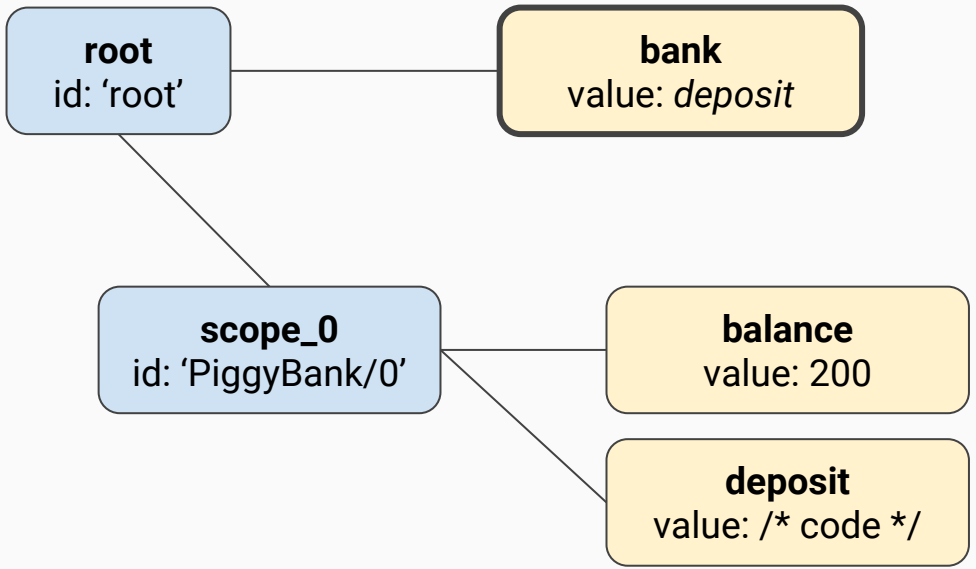
Implementation - Serialization


```
function PiggyBank(){
    var balance = 0
    var deposit = function(amount){
        balance += amount
    }
    return deposit
}
var bank = PiggyBank()
```



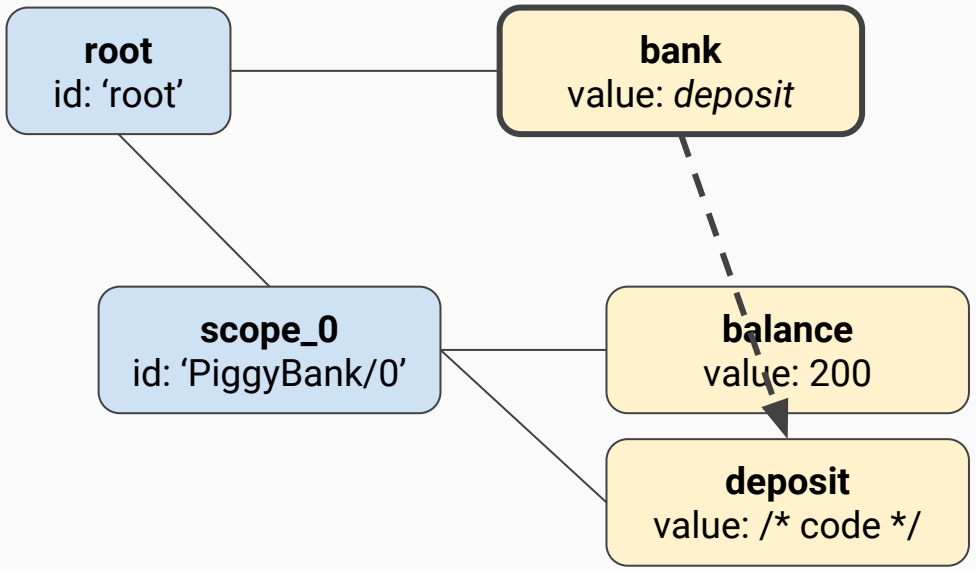
Implementation - Serialization


```
function PiggyBank(){
    var balance = 0
    var deposit = function(amount){
        balance += amount
    }
    return deposit
}
var bank = PiggyBank()
```



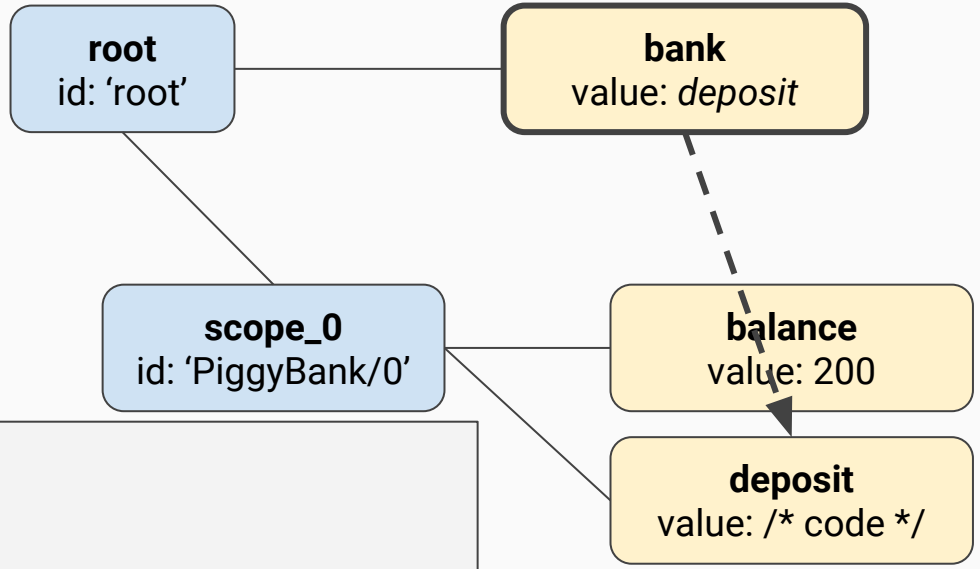
Implementation - Serialization


```
function PiggyBank(){
    var balance = 0
    var deposit = function(amount){
        balance += amount
    }
    return deposit
}
var bank = PiggyBank()
```



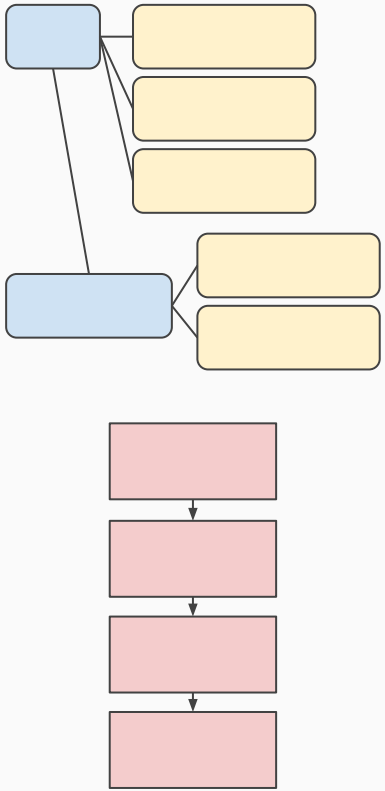
Implementation - Serialization


```
function PiggyBank(){
    var balance = 0
    var deposit = function(amount){
        balance += amount
    }
    return deposit
}
var bank = PiggyBank()
```



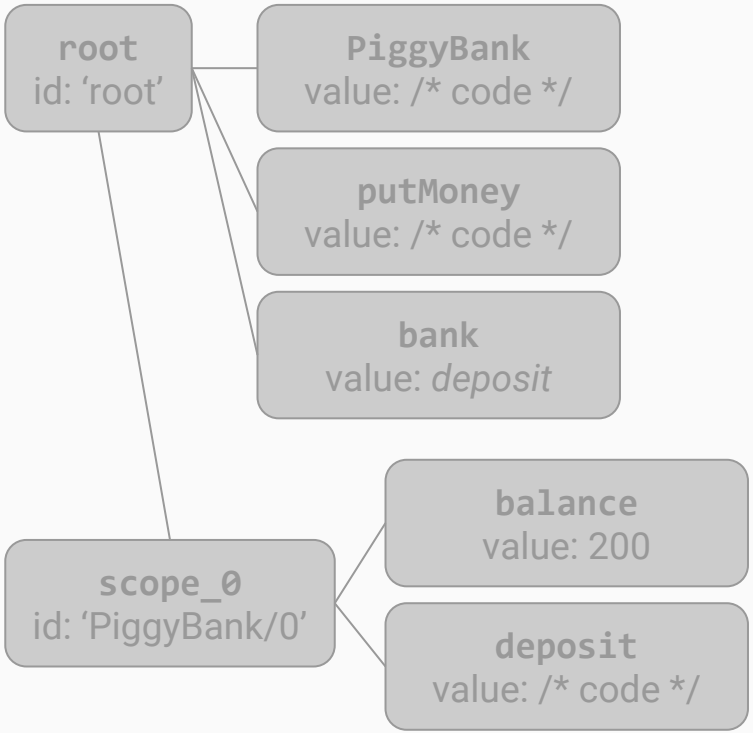

```
'root': {
  'vars': {
    'bank': {
      'type': 'function-reference',
      'value': 'PiggyBank/0.deposit'
    }
  }
}
```

Implementation - Serialization

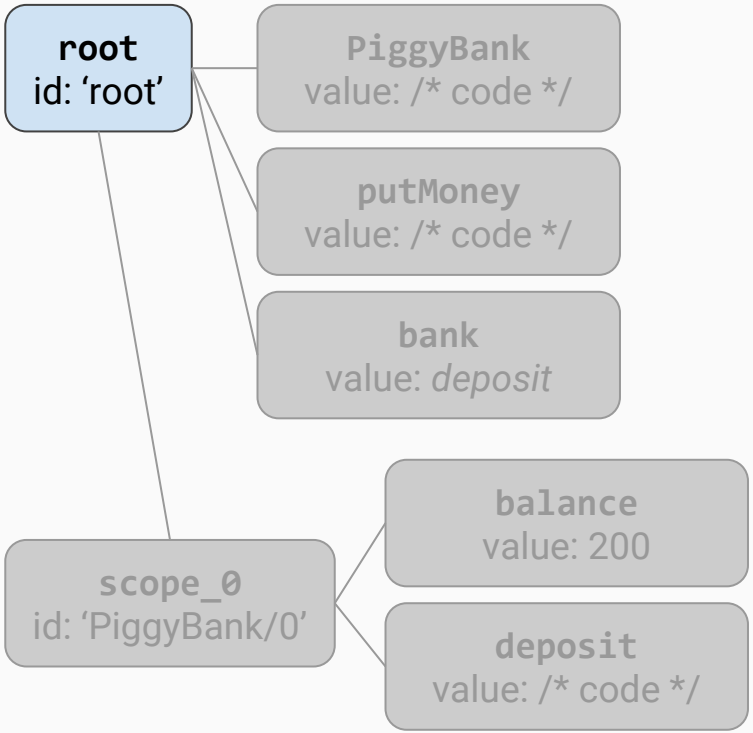


```
'root': {
  'vars': {
    'bank': { /* truncated */ }
  },
  'children': {
    'PiggyBank/0': {
      'vars': { /* truncated */ }
    }
  }
  'timers': {
    '0': {
      'type': 'Interval',
      'callback': 'putMoney',
      'delay': 1000,
      'remaining': 500
    }
}
```

Implementation - Restoration

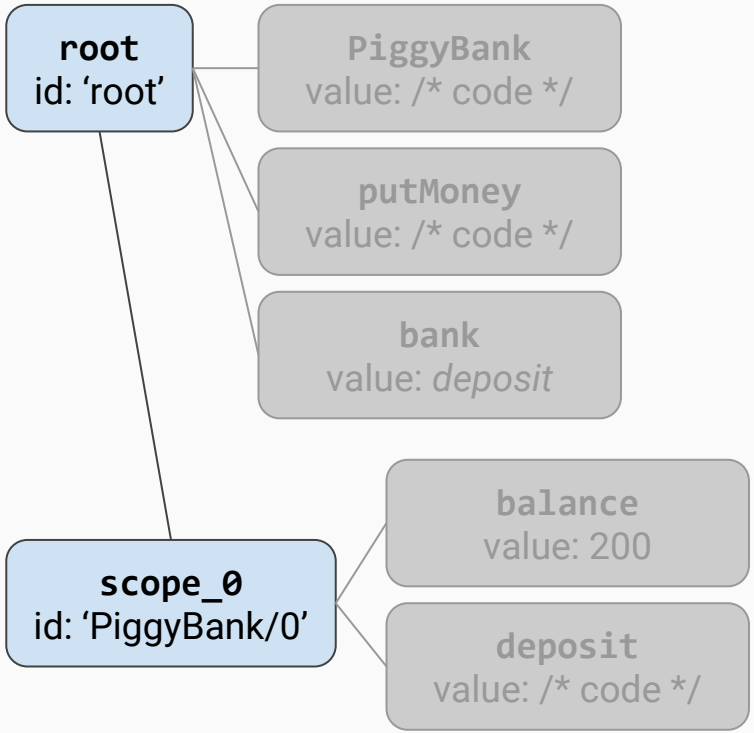


Implementation - Restoration



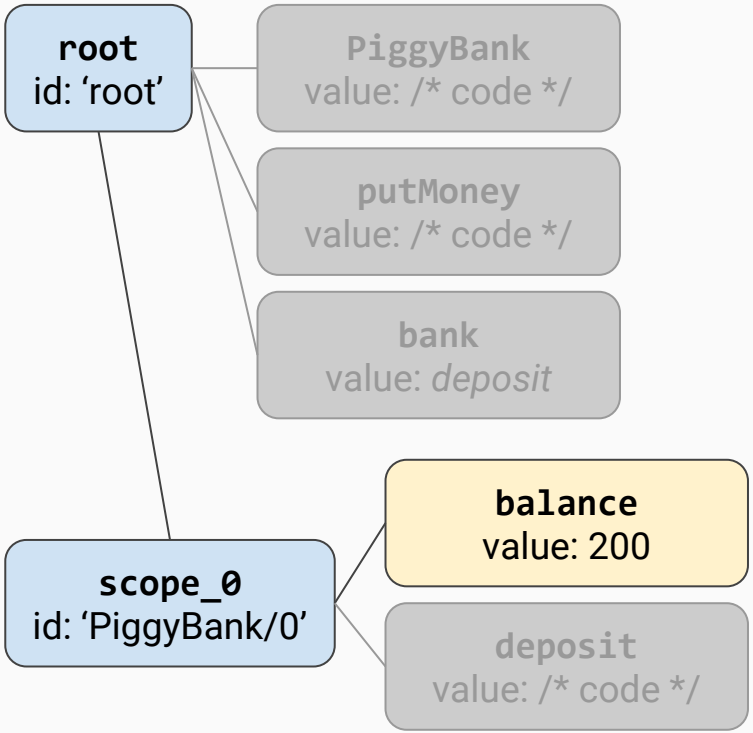

```
var root = new Scope()
```

Implementation - Restoration



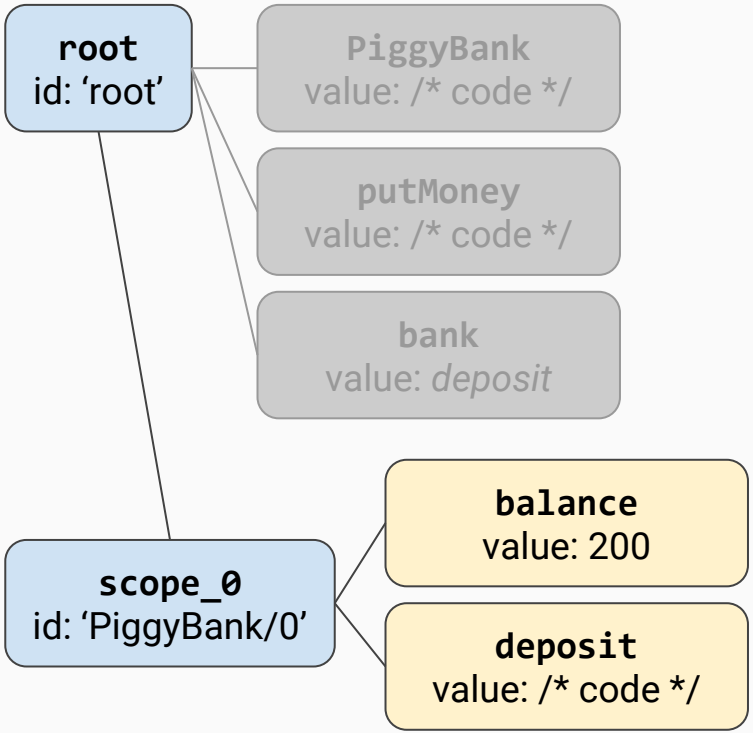

```
var root = new Scope()
(function PiggyBank_0(){
   var scope_0 = new Scope(root)









})()
```

Implementation - Restoration



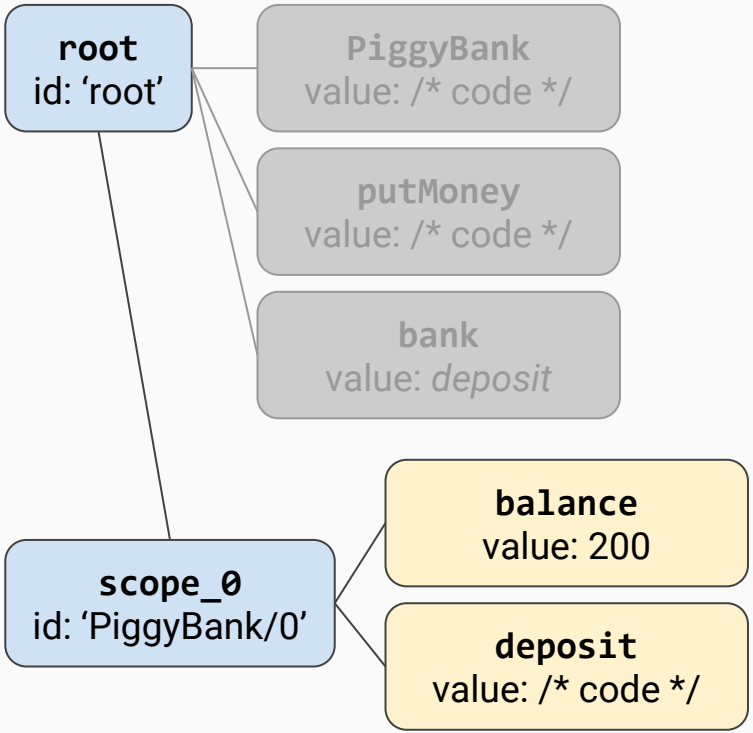

```
var root = new Scope()
(function PiggyBank_0(){
   var scope_0 = new Scope(root)
   var balance = 200
   scope_0.vars.balance = balance




})()
```

Implementation - Restoration



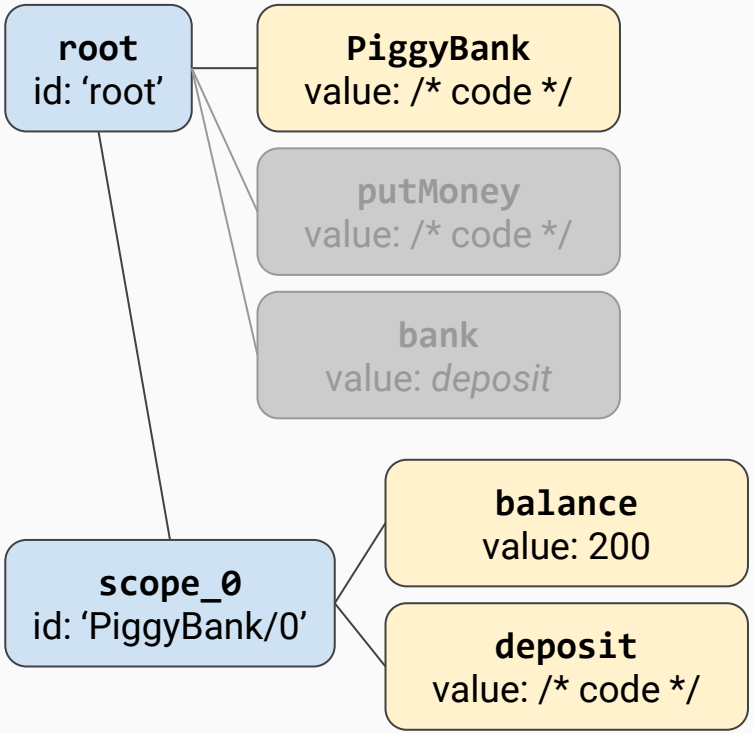

```
var root = new Scope()
(function PiggyBank_0(){
   var scope_0 = new Scope(root)
   var balance = 200
   scope_0.vars.balance = balance
   var deposit = function(amount){
      /* Original code */
   }
   scope_0.addFunction(deposit)
})()
```

Implementation - Restoration



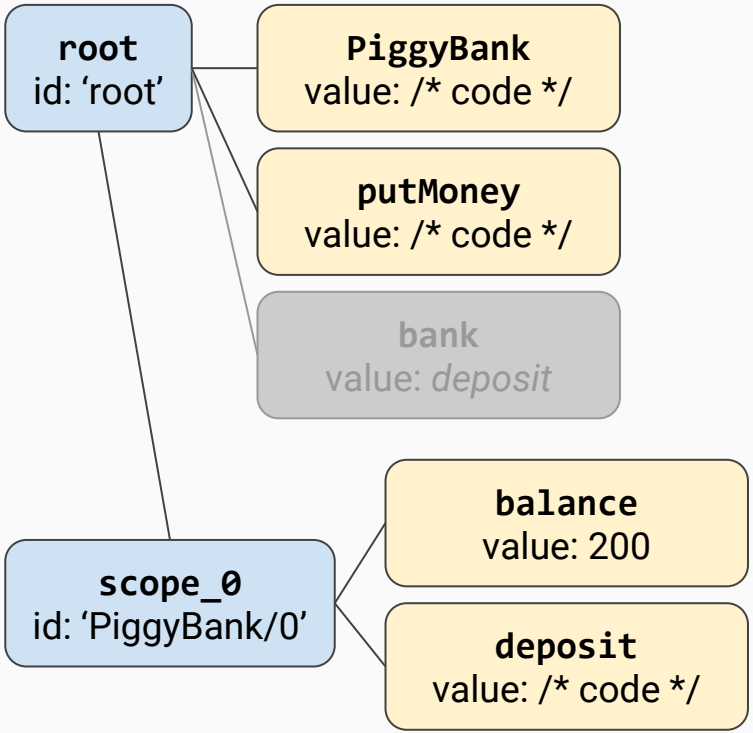

```
var root = new Scope()
(function PiggyBank_0(){
   /* truncated */
})()
```

Implementation - Restoration



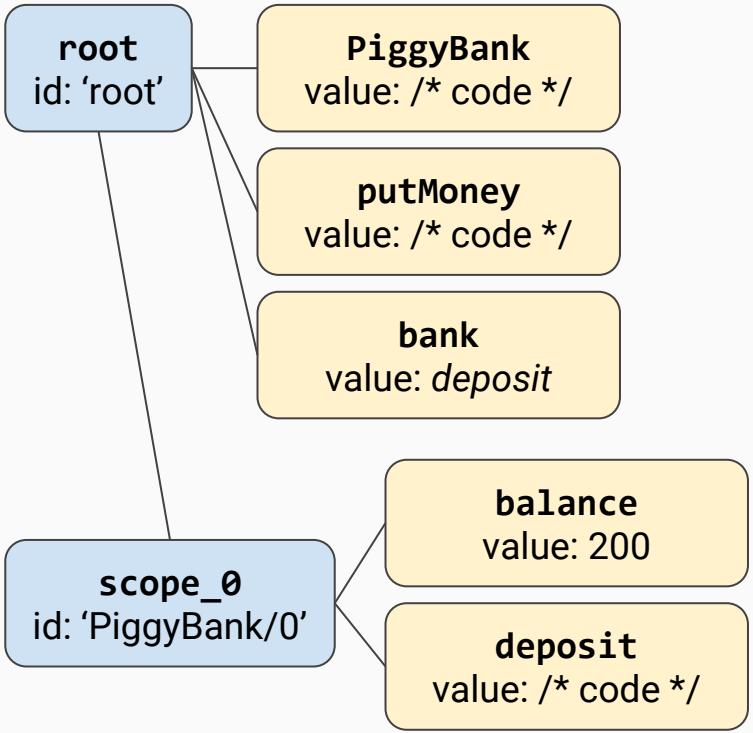

```
var root = new Scope()
(function PiggyBank_0(){
   /* truncated */
})()
function PiggyBank(){
   /* Original code */
}
root.addFunction(PiggyBank)
```

Implementation - Restoration



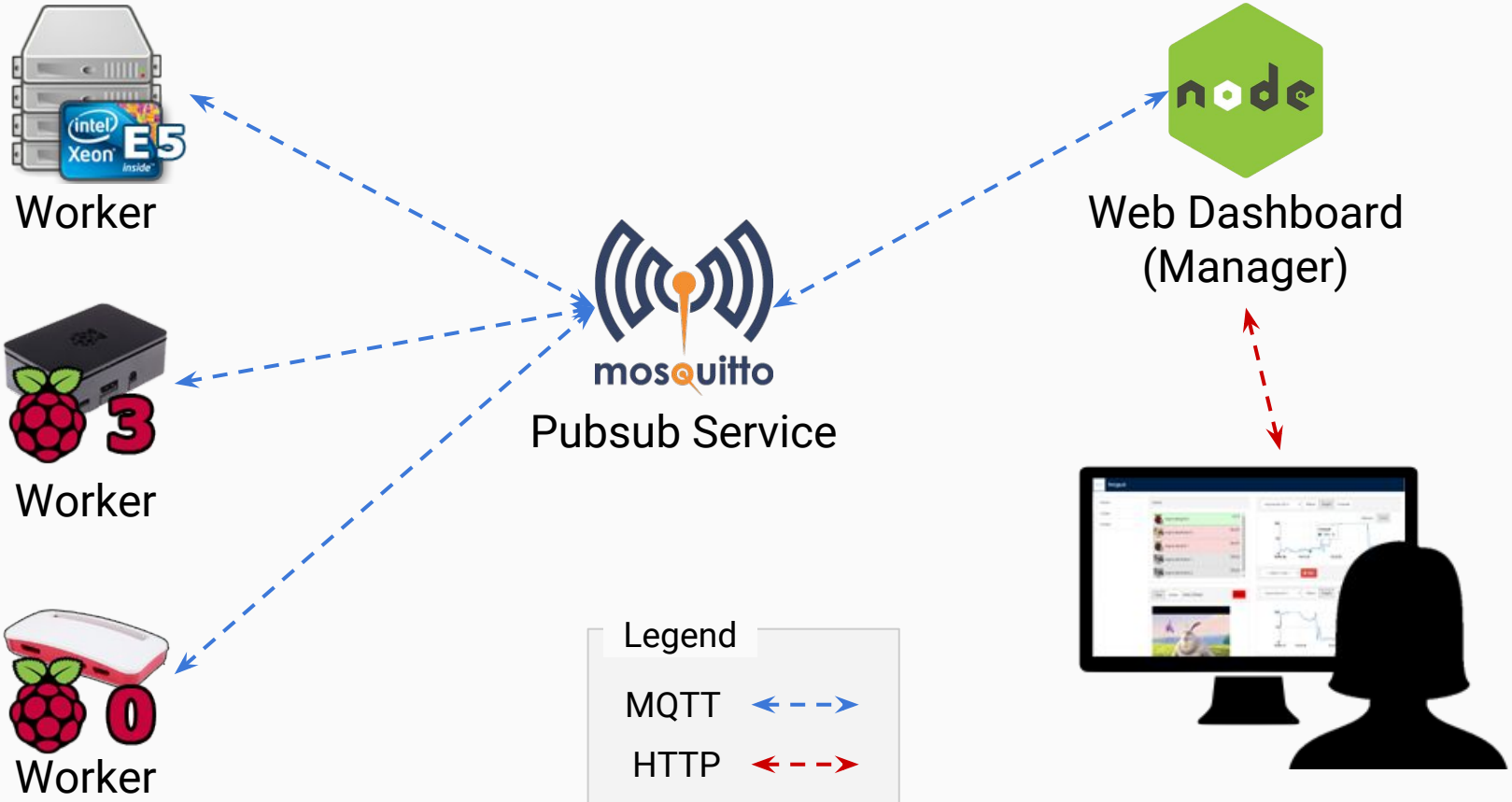

```
var root = new Scope()
(function PiggyBank_0(){
   /* truncated */
})()
function PiggyBank(){
   /* Original code */
}
root.addFunction(PiggyBank)
root.addFunction(function putMoney(){
   /* Original code */
})
```

Implementation - Restoration

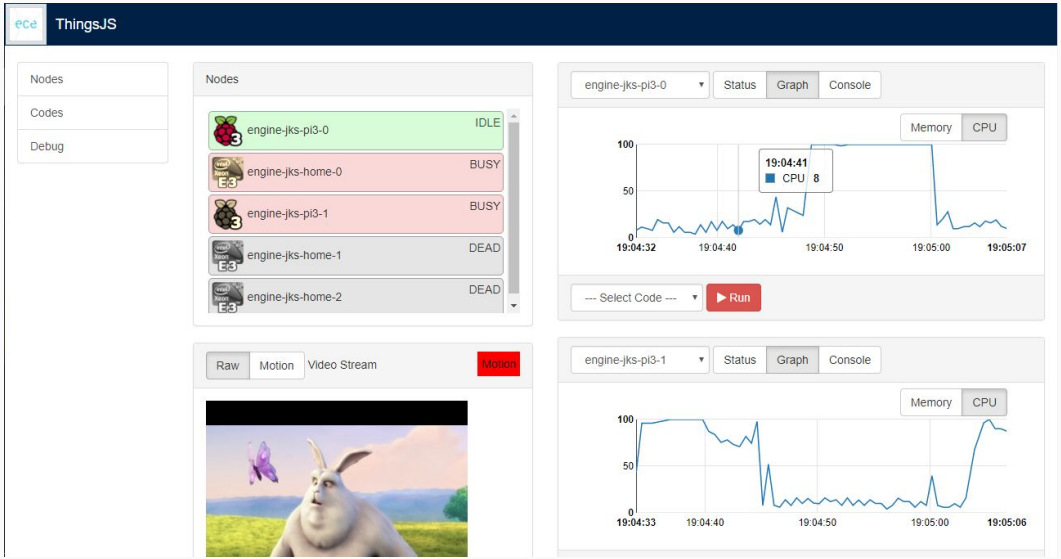



```
var root = new Scope()
(function PiggyBank_0(){
   /* truncated */
})()
function PiggyBank(){
   /* Original code */
}
root.addFunction(PiggyBank)
root.addFunction(function putMoney(){
   /* Original code */
})
var bank =
root.getFunction('PiggyBank/0.deposit')
root.vars.bank = bank
```

Experiment


Worker
Worker
Worker
mosquitto
Pubsub Service
node
Web Dashboard
(Manager)
Legend
MQTT
HTTP

Project available on Github



github.com/karthikp-ubc/ThingsJS

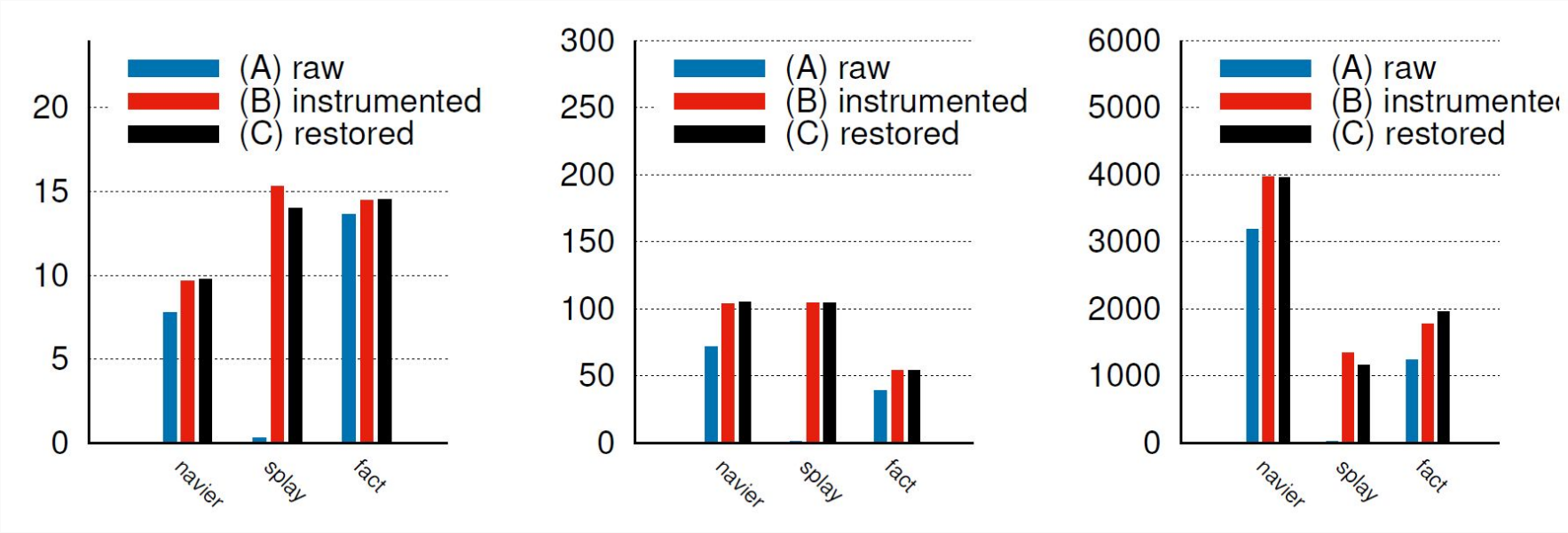
Results

Benchmarks

- *Chrome Octane* Suite
  - NavierStokes - **CPU intensive**
  - Splay - **Memory intensive**
- Factorial
- Regulator

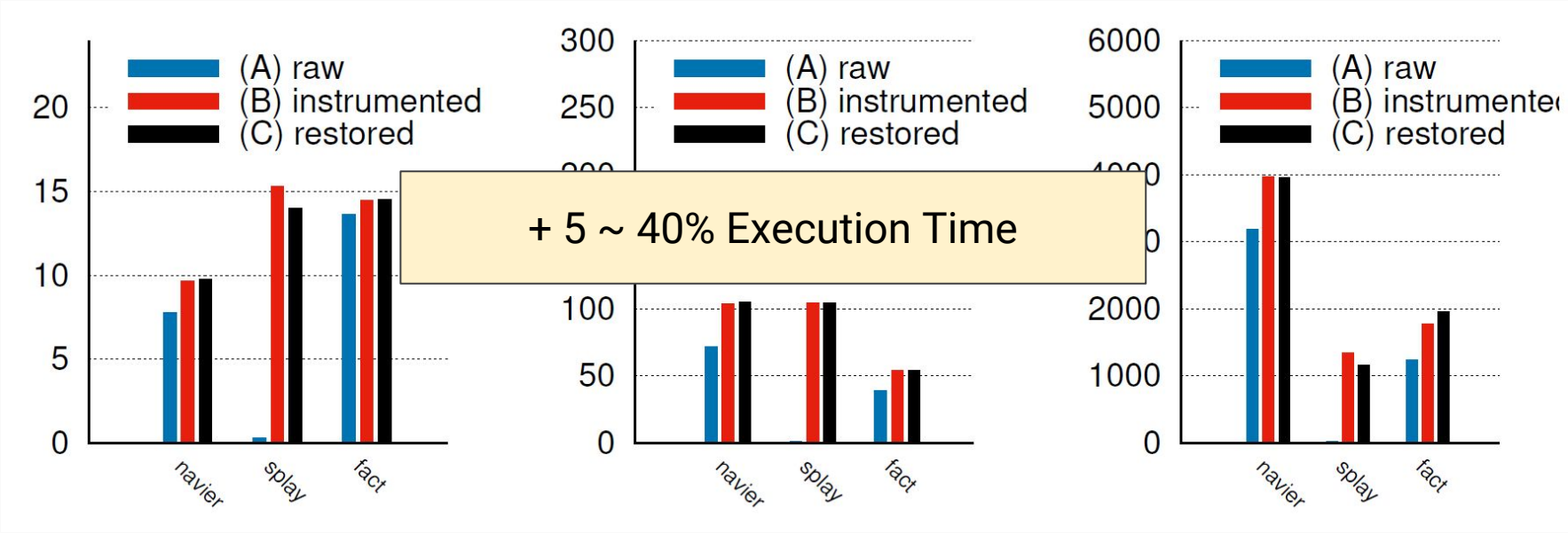
Results

Execution Time



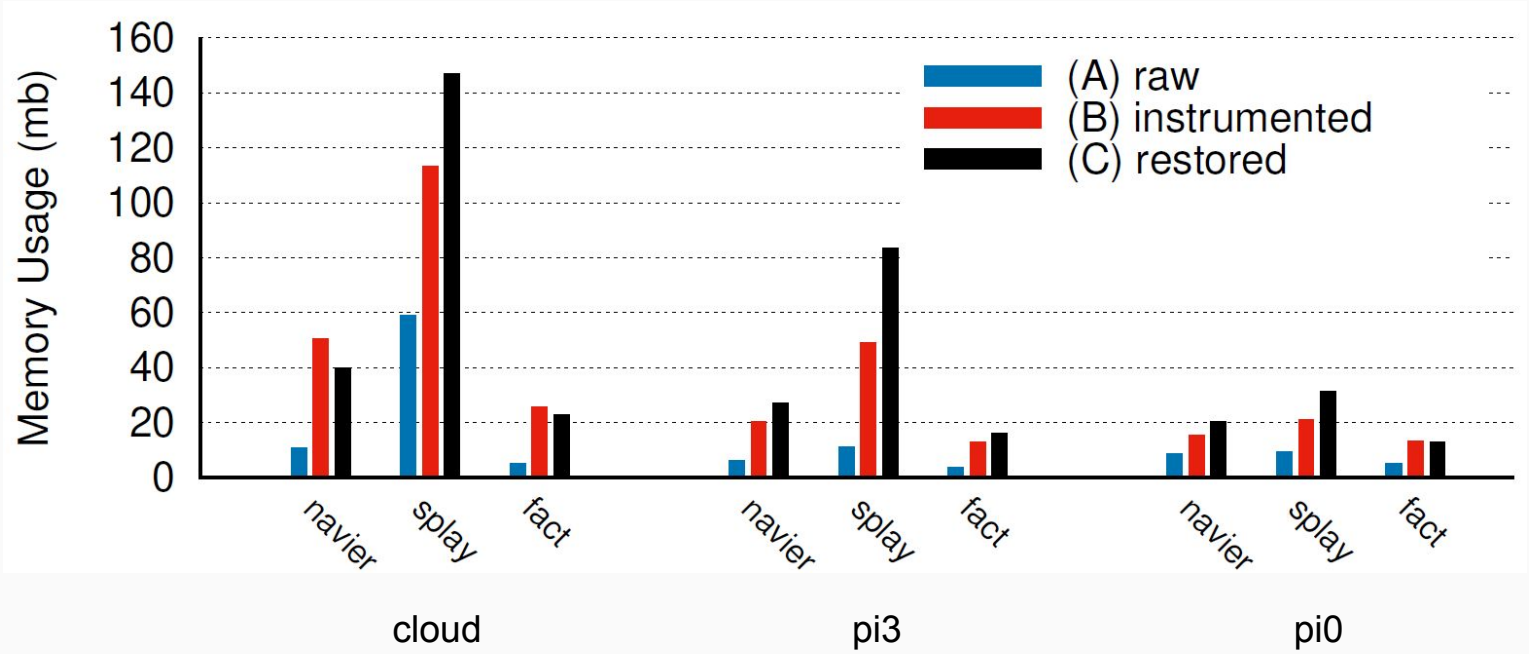
Results

Execution Time



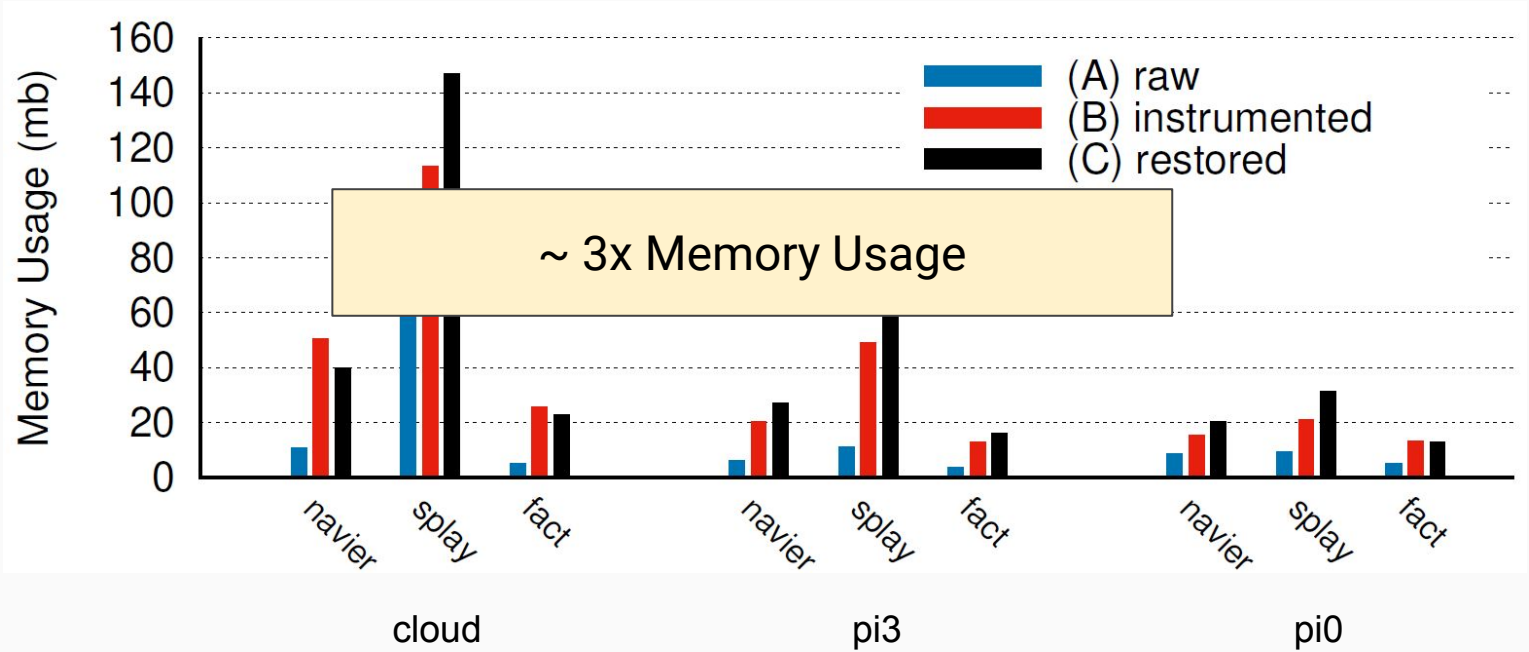
Results

Memory Usage

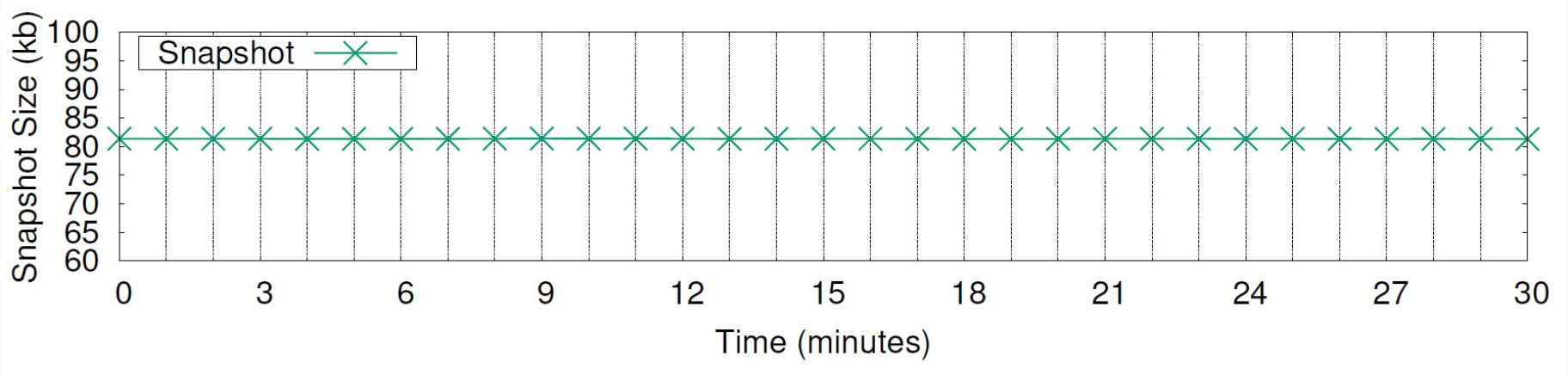


Results

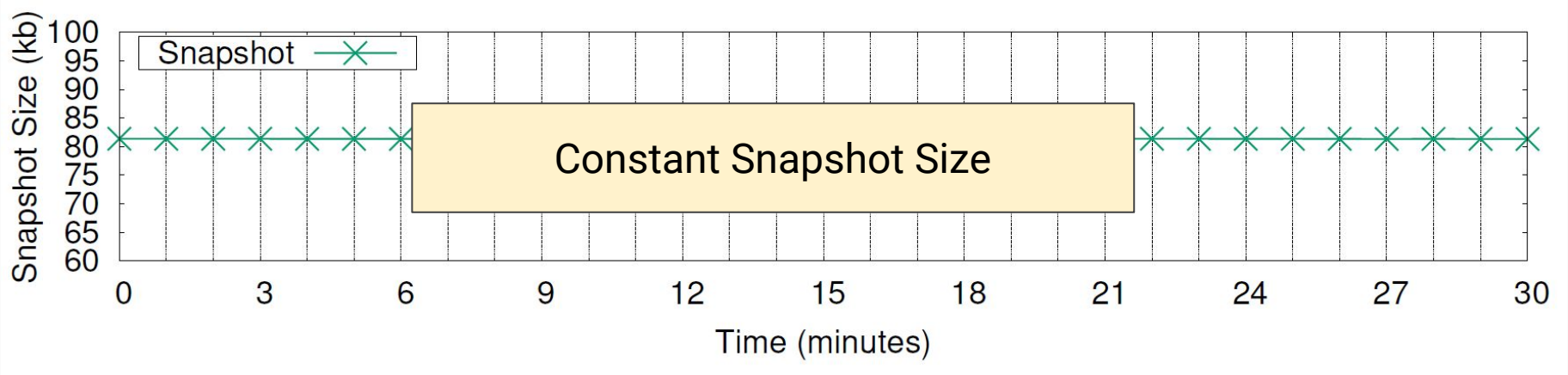
Memory Usage



Multi-hop Behaviour



Multi-hop Behaviour



Future Work

1. Optimization
   a. Memory usage
   b. CPU usage
2. Fault-tolerance
   a. Infinite blocking loops/recursion
   b. Checkpointing
   c. Real-time state streaming
3. Decentralized scheduling

tl;dr

We provide a high-level migration framework for JavaScript programs

- Stateful applications
- Platform-independent
- No VM modification

thingsjs.juliengs.com
kumseok@ece.ubc.ca