

Swift-CowBox

Easy Copy-on-Write Semantics for Swift Structs

Rick van Voorden • Los Angeles • California



Swift-CowBox

Easy Copy-on-Write Semantics for Swift Structs

- Background
- Demo
- Case-Study
- Tips and Tricks

Background



Background

Reference Semantics and Value Semantics

```
class Person {  
    var location: String  
}
```

```
let p1 = Person(  
    location: "Los Angeles"  
)  
var p2 = p1  
p2.location = "Paris"  
print(p1.location)  
// Prints "Paris"
```

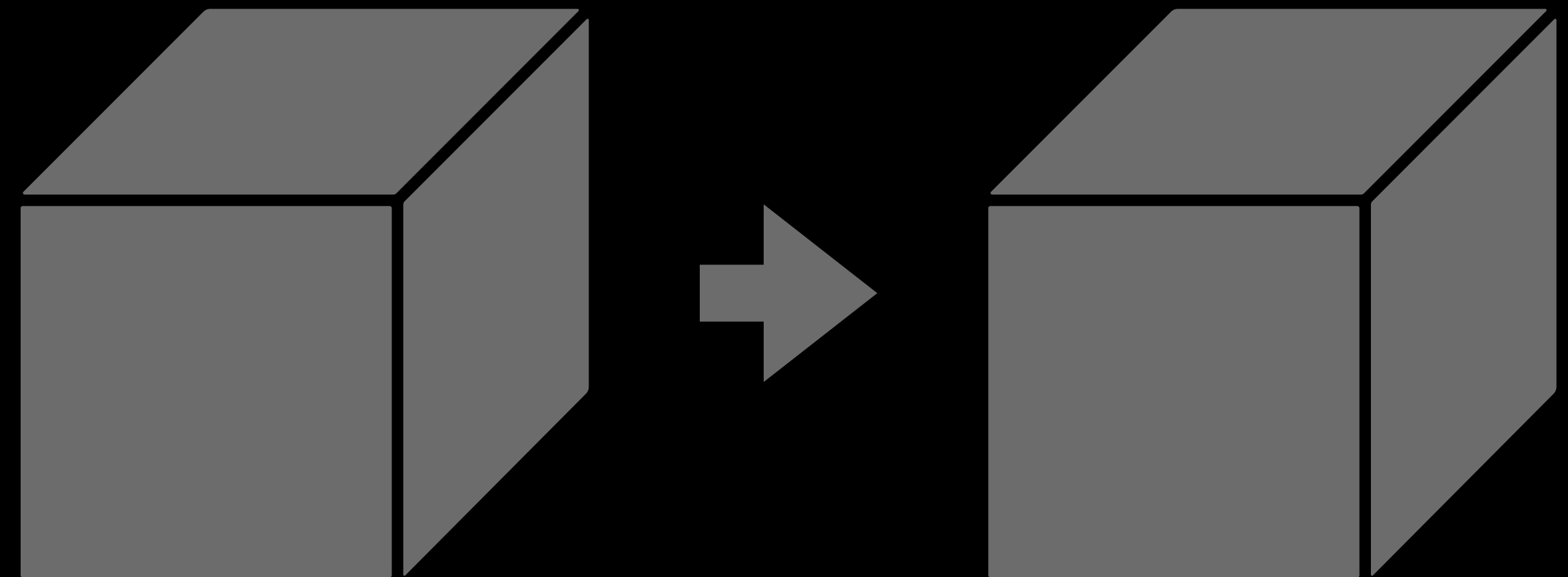
```
struct Person {  
    var location: String  
}
```

```
let p1 = Person(  
    location: "Los Angeles"  
)  
var p2 = p1  
p2.location = "Paris"  
print(p1.location)  
// Prints "Los Angeles"
```

Memory

Value Semantics

```
struct Container {  
    var a: Int64  
    var b: Int64  
    var c: Int64  
    var d: Int64  
    var e: Int64  
    var f: Int64  
    var g: Int64  
    var h: Int64  
    var i: Int64  
    var j: Int64  
}
```



80 Bytes

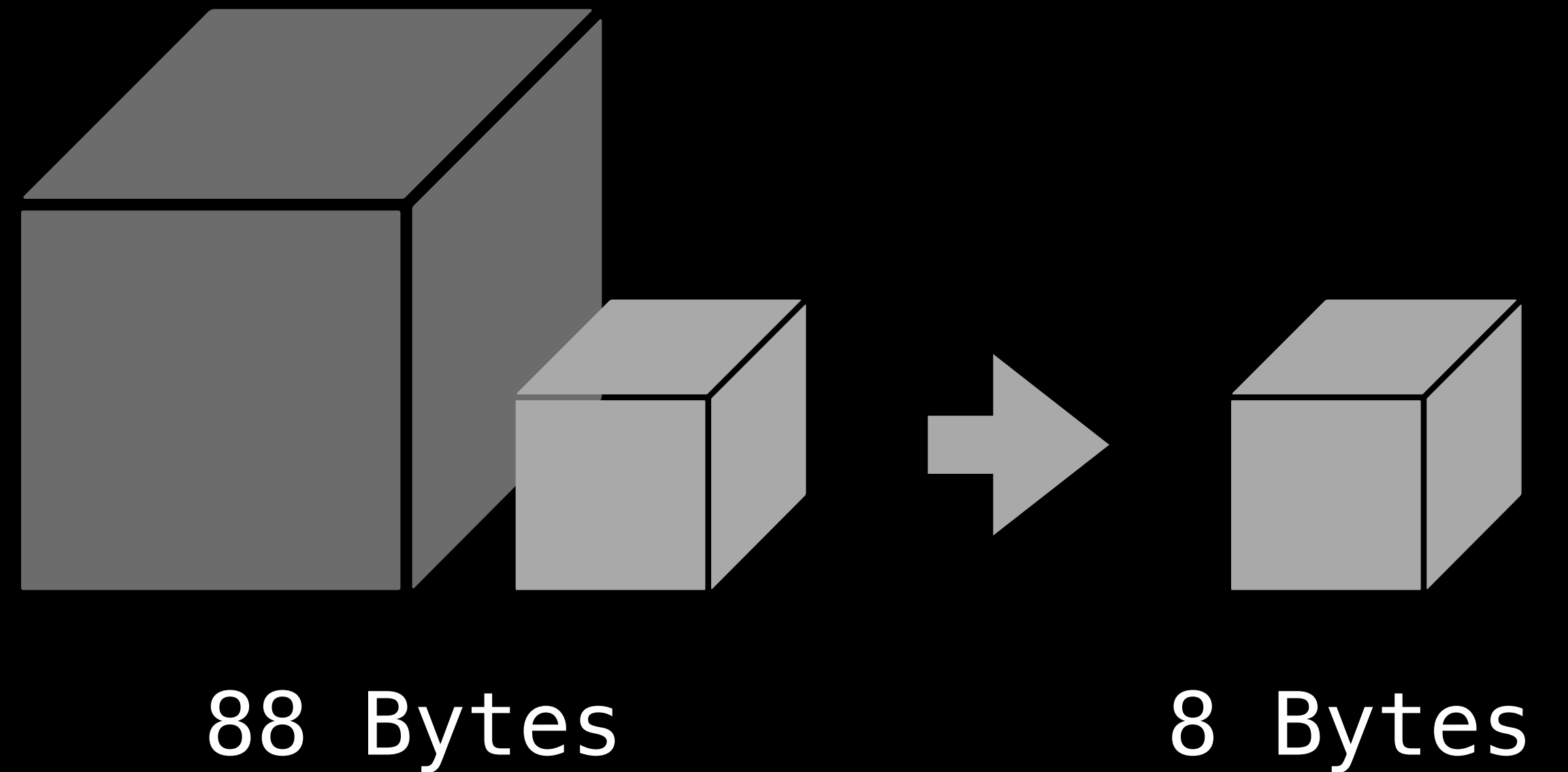
80 Bytes

$$M(n) = 80n$$

Memory

Reference Semantics

```
class Container {  
  var a: Int64  
  var b: Int64  
  var c: Int64  
  var d: Int64  
  var e: Int64  
  var f: Int64  
  var g: Int64  
  var h: Int64  
  var i: Int64  
  var j: Int64  
}
```



$$M(n) = 8n + 80$$

Memory

Reference Semantics and Value Semantics

```
class Container {  
  var a: Int64  
  var b: Int64  
  var c: Int64  
  var d: Int64  
  var e: Int64  
  var f: Int64  
  var g: Int64  
  var h: Int64  
  var i: Int64  
  var j: Int64  
}
```

$$M(n) = 8n + 80$$

```
struct Container {  
  var a: Int64  
  var b: Int64  
  var c: Int64  
  var d: Int64  
  var e: Int64  
  var f: Int64  
  var g: Int64  
  var h: Int64  
  var i: Int64  
  var j: Int64  
}
```

$$M(n) = 80n$$

Copy-on-Write

Reference Semantics... and Value Semantics?

- Interface follows value semantics
 - Local reasoning
- Implementation follows reference semantics
 - Shared storage
- Array and Dictionary

Copy-on-Write

By-Hand

```
struct Person {  
    let id: String  
    var location: String  
}
```

Copy-on-Write

By-Hand

```
extension Person {  
    private final class _Storage {  
        let id: String  
        var location: String  
        init(id: String, location: String) {  
            self.id = id  
            self.location = location  
        }  
        func copy() -> _Storage {  
            _Storage(id: self.id, location: self.location)  
        }  
    }  
}
```

Copy-on-Write

By-Hand... and Hand

```
struct Person {  
    private var _storage: _Storage  
  
    init(id: String, location: String) {  
        self._storage = _Storage(id: id, location: location)  
    }  
}
```

Copy-on-Write

By-Hand... and Hand... and Hand

```
extension Person {  
    var id: String {  
        get {  
            self._storage.id  
        }  
    }  
}
```

Copy-on-Write

By-Hand... and Hand... and Hand... and Hand

```
extension Person {  
    var location: String {  
        get {  
            self._storage.location  
        }  
        set {  
            if isKnownUniquelyReferenced(&self._storage) == false {  
                self._storage = self._storage.copy()  
            }  
            self._storage.location = newValue  
        }  
    }  
}
```

Copy-on-Write

By-Hand... and Hand... and Hand... and Hand

```
struct Person {
    private var _storage: _Storage

    init(id: String, location: String) {
        self._storage = _Storage(
            id: id,
            location: location
        )
    }
}

extension Person {
    var location: String {
        get {
            self._storage.location
        }
        set {
            if !isKnownUniquelyReferenced(&self._storage) {
                self._storage = self._storage.copy()
            }
            self._storage.location = newValue
        }
    }
}
```

```
extension Person {
    private final class _Storage {
        let id: String
        var location: String
        init(id: String, location: String) {
            self.id = id
            self.location = location
        }
        func copy() -> _Storage {
            _Storage(
                id: self.id,
                location: self.location
            )
        }
    }
}

extension Person {
    var id: String {
        get {
            self._storage.id
        }
    }
}
```

Copy-on-Write

TODO

```
struct Person {  
    // TODO: MOVE ID TO COPY ON WRITE  
    let id: String  
    // TODO: MOVE LOCATION TO COPY ON WRITE  
    var location: String  
}
```

Demo

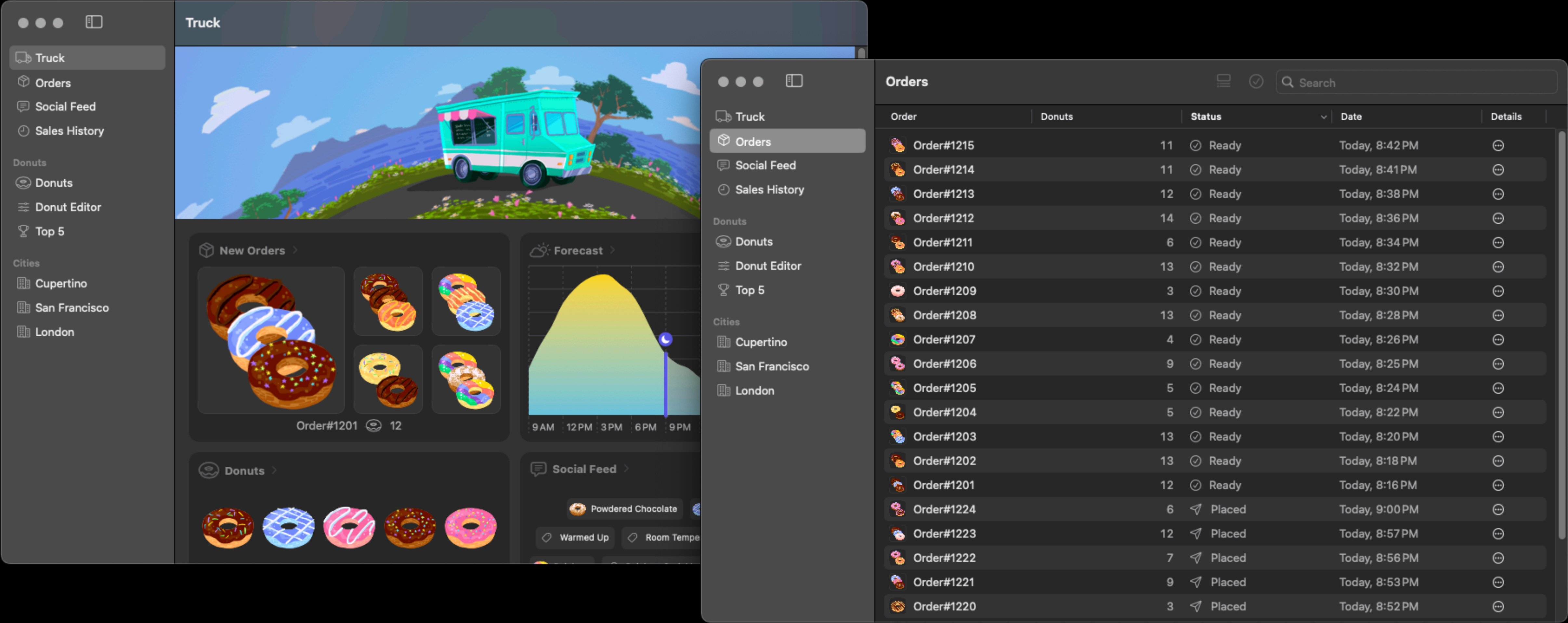


Case-Study



Case-Study

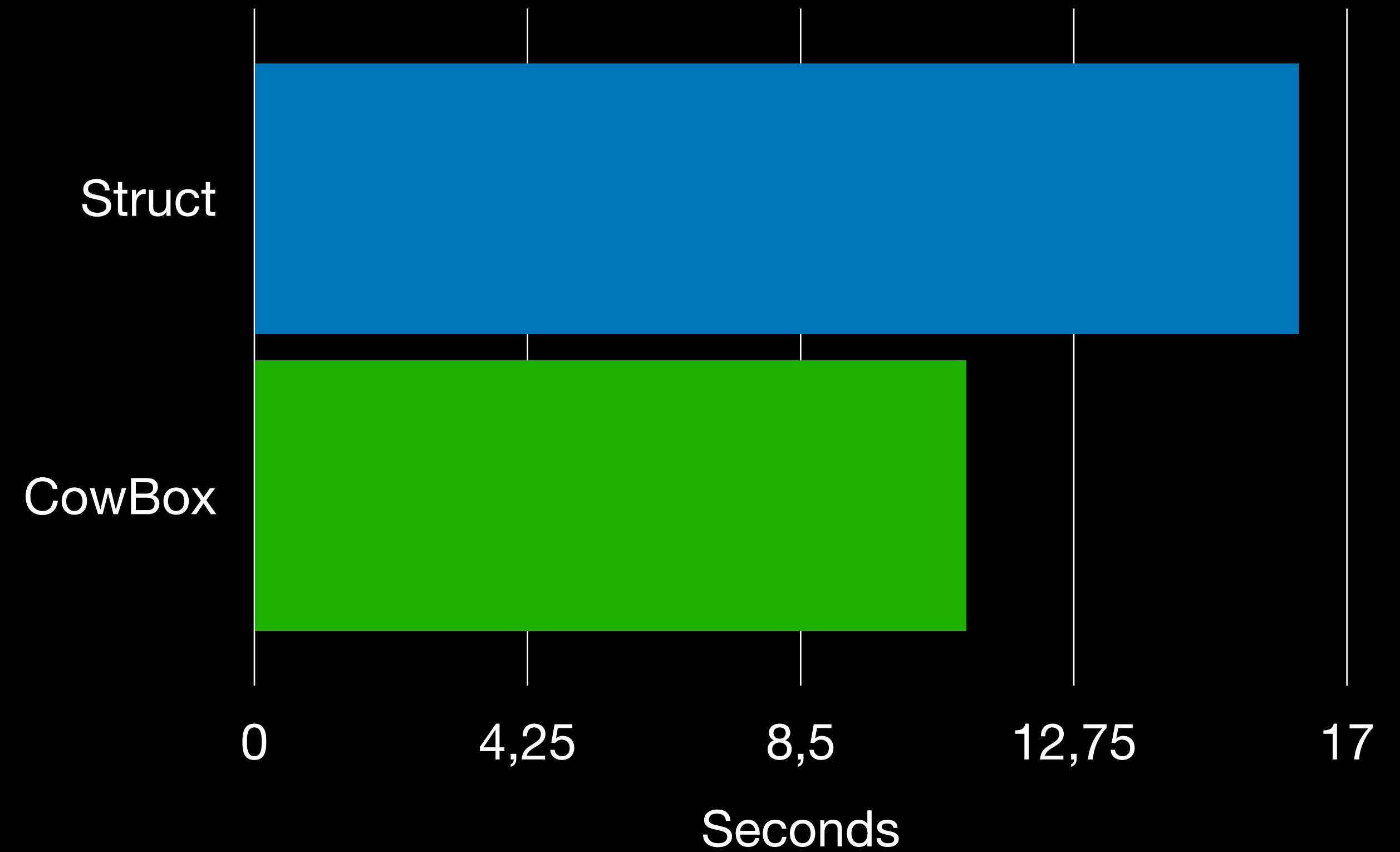
Food Truck



Measurements

Core Animation Commits

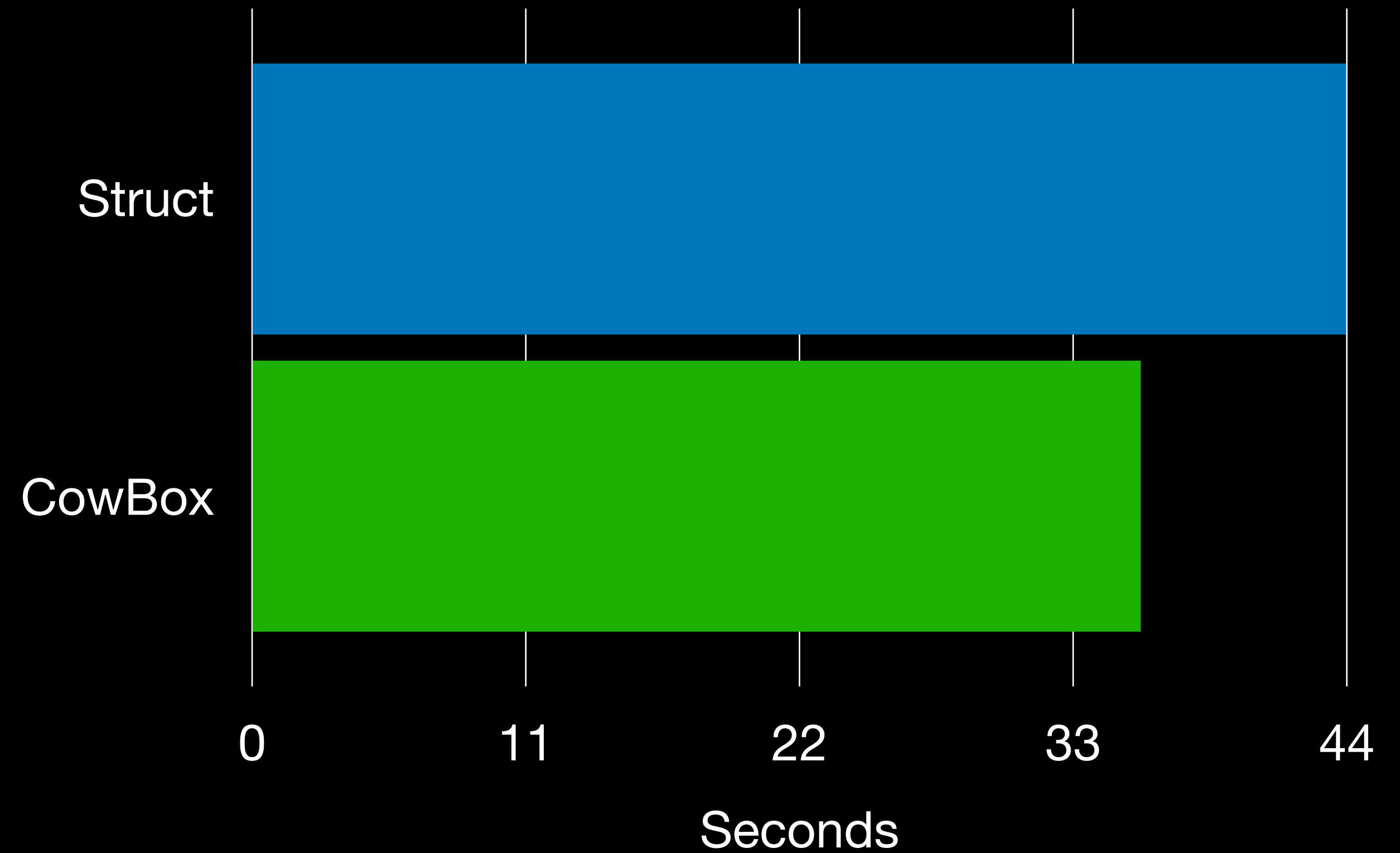
- Order Struct
 - 16.25 Seconds
- Order CowBox
 - 11.07 Seconds
- 31 Percent Reduction



Measurements

Hangs and Hitches

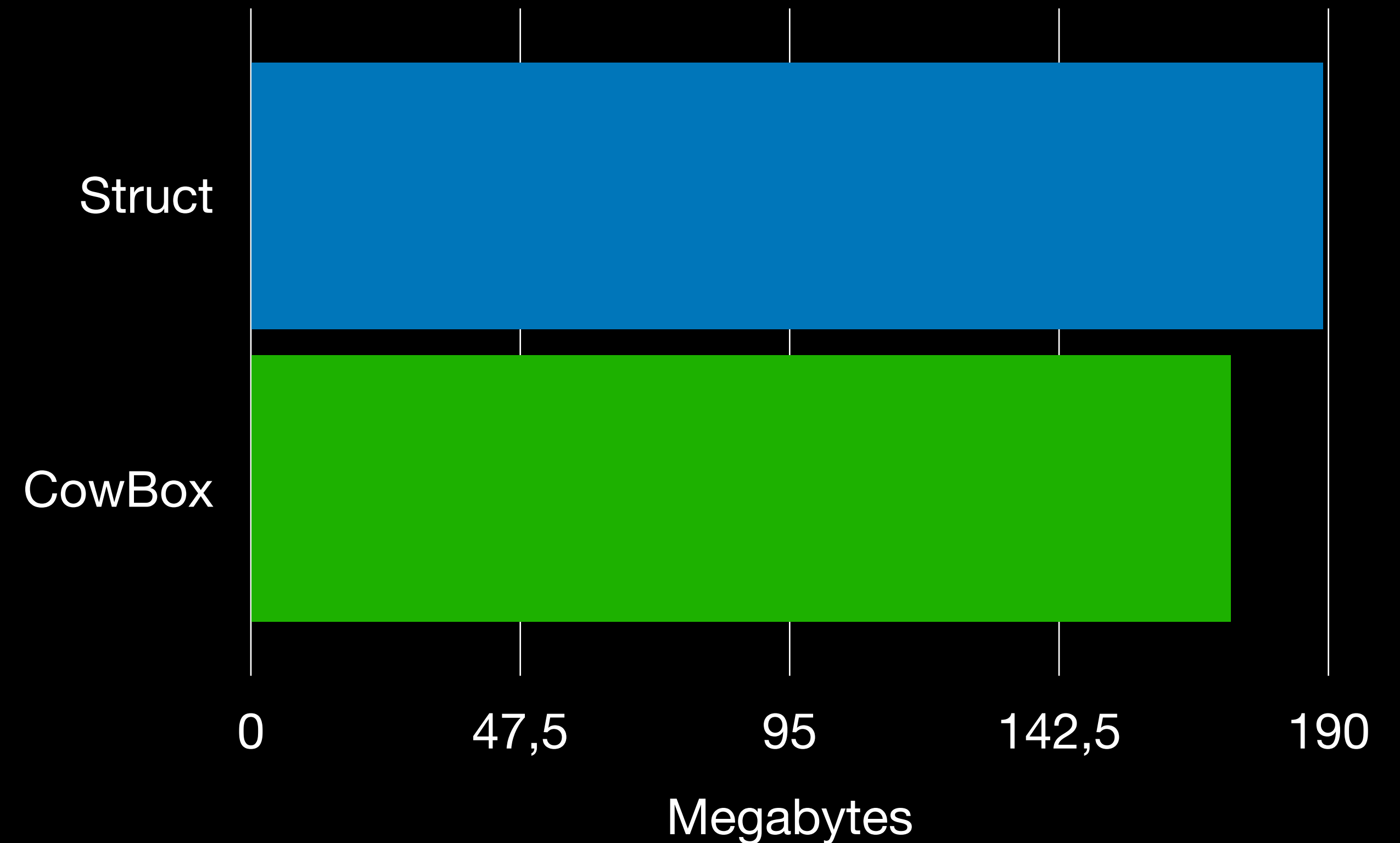
- Order Struct
 - 43.99 Seconds
- Order CowBox
 - 35.72 Seconds
- 18 Percent Reduction



Measurements

Persistent Memory

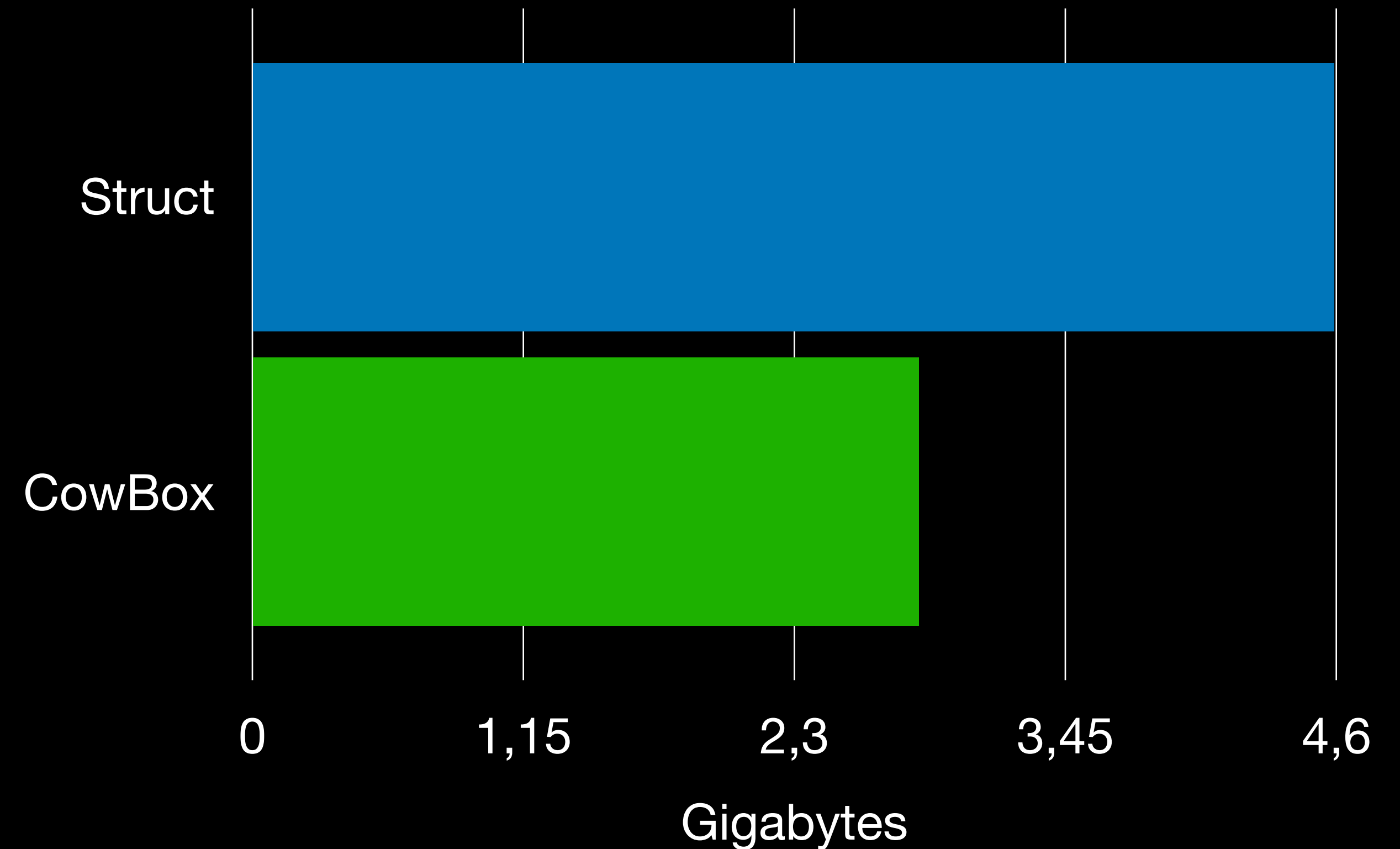
- Order Struct
 - 189.19 Megabytes
- Order CowBox
 - 172.82 Megabytes
- 8 Percent Reduction



Measurements

Total Memory

- Order Struct
 - 4.59 Gigabytes
- Order CowBox
 - 2.83 Gigabytes
- 38 Percent Reduction



import CowBox

Tips and Tricks



Tips and Tricks

When might performance *not* improve?

- Compile Time
- Run Time
 - CPU
 - Memory
- Accidental Quadratics

Tips and Tricks

Getting Started

- Start with a baseline measurement
 - Measure memory and CPU
 - Ordo One Benchmarks
 - Apple Instruments

Tips and Tricks

Getting Started

- Choose one type to migrate
 - Many bytes of storage
 - Copied many times
 - Checked for equality
- Measure your changes

```
struct Person {
    private var _storage: _Storage

    init(id: String, location: String) {
        self._storage = Storage(
            id: id,
            location: location
        )
    }
}

extension Person {
    private final class _Storage {
        let id: String
        var location: String
        init(id: String, location: String) {
            self.id = id
            self.location = location
        }
        func copy() -> _Storage {
            _Storage(
                id: self.id,
                location: self.location
            )
        }
    }
}

extension Person {
    var location: String {
        get {
            self._storage.location
        }
        set {
            if !isKnownUniquelyReferenced(&self._storage) {
                self._storage = self._storage.copy()
            }
            self._storage.location = newValue
        }
    }
}
```



@CowBox

github.com/swift-cowbox



github.com/vanvoorden



Acknowledgements

Special Thanks

- DzuyAn Nguyen
- Jeff Ort
- Knott Wittawat
- Bill Fisher

References

Further Reading

- James Dempsey • Value and Reference Types in Swift
- Karoy Lorentey • Optimizing Swift Collections
- Ben Cohen • Fast Safe Mutable State
- Johannes Weiss • High-Performance Systems in Swift
- Cory Benfield • High-Performance Systems in Swift
- Jared Khan • Swift's Copy-on-write Optimisation

Merci!

github.com/swift-cowbox

