Fixpoint: A Functional Operating System

Yuhan Deng, Angela Montemayor, Keith Winstein
Stanford University
Three Parables of Modern Computing

• Parable 1: The Sloppy Data Scientist
• Parable 2: The Anxious Cloud Storage Company
• Parable 3: The Inefficient Serverless Platform
Parable 1: The Sloppy Data Scientist

“Yuhan, how did you generate that graph? I’d love to reproduce your result.”

Sure! Give me one second.

```bash
$ git clone https://github.com/yhdengh/myproject.git
Cloning into 'myproject'...
$ cd myproject/
~/myproject$ ls test-result/
graph.png res1 res2 result-new result-final result-final-new result-final-final
    test-script
~/myproject$ ls test-result/test-script/
draw.py test1.py test2.py helper.py
~/myproject$ oops
```

Ummm. I guess…
~/myproject$ git add computations-that-generate-graph.png
~/myproject$ git commit "Add computational relationship for graph.png"
~/myproject$ git push
Parable 2: The Anxious Cloud Storage Company

2022/1/1

User

→ Lossless Compress

→ Decompress

→ Same as the original image

Storage

2027/1/1

What if decompression is NOT deterministic

User

→&nbsp;.lossless compress →&nbsp;decompress →&nbsp;same as the original image

Storage
However…
- where the program is placed matters in terms of the total execution time
Three Parables of Modern Computing

- Parable 1: The Sloppy Data Scientist
- Parable 2: The Anxious Cloud Storage Company
- Parable 3: The Inefficient Serverless Platform
Takeaway

Separating processes into deterministic computations and pre-declared dependencies helps to solve these issues:

- Parable 1: The Sloppy Data Scientist
- Parable 2: The Anxious Cloud Storage Company
- Parable 3: The Inefficient Serverless Platform
Outline

• Introduction
• Our proposed solution: Fixpoint
• How is Fixpoint implemented
• Demo
• Open Questions
• Conclusion
Outline

• Introduction
• Our proposed solution: Fixpoint
• How is Fixpoint implemented
• Demo
• Open Questions
• Conclusion
Fixpoint separates a process into a deterministic computation and a list of pre-declared dependencies

Processes are essentially opaque boxes to the OS

Processes are semi-opaque. Each process expose to the OS a deterministic computation and a list of pre-declared dependencies.
Fixpoint separates a process into a deterministic computation and a list of pre-declared dependencies.

- Deterministic computation (function)
  - Represented as a WebAssembly module

- List of dependencies (List of inputs)
  - Content-addressed storage
  - All dependencies have to be ready before the computation starts
Fixpoint expresses computational relationship between objects

Computational relationship between inputs and outputs of deterministic functions

A large program can be disassembled into more fine-grained computation steps
Three Parables of Modern Computing

• Parable 1: The Sloppy Data Scientist
• Parable 2: The Anxious Cloud Storage Company
• Parable 3: The Inefficient Serverless Platform
Parable 1: The Sloppy Data Scientist

```bash
~/myproject$ git add computation-relation-for-graph.png
~/myproject$ git commit "Add computational relationship for graph.png"
~/myproject$ git push
~/myproject$ cd test-result
~/myproject/test-result$ git why graph.png
test-scripts/draw.py result-final-final
~/myproject/test-result$ git why result-final-final
test-scripts/test1.py ../src/input-data
~/myproject$ git rebase ../src/another-input
Executing "test-scripts/test1.py ../src/another-input" ...
Executing "test-scripts/draw.py result-final-final" ...
~/myproject$ git add another-computation-step
Executing "another-computation-step graph.png" ...
```
**Parable 2: The Anxious Cloud Storage Company**

2022/1/1

User

Lossless Compress

Decompress

Same as the original image

Storage

![](image1.png)

2027/1/1

Decompress

Same as the original image

What if decompression is NOT deterministic

IT MUST BE

Hey you goin’ to sleep?

Yes, now shut up

😀
Parable 3: The Inefficient Serverless Platform

- Fine-grained computations that can be executed anywhere
- Pre-declared dependencies

- Makes placement decision with info about where dependencies are coming from

- Charges by computation
- Provides certificate that a computation is completed correctly

Charge by execution time

User

Serverless Computing Platform
Outline

• Introduction
• Our proposed solution: Fixpoint
• How is Fixpoint implemented
• Demo
• Open Questions
• Conclusion
Blob – stores the contents of a single file  
Tree – contains references to other blobs or subtrees  
Commit – refers to another tree object and some other information  
Encode – represents a pair of computation and a list of dependencies  
Thunk – refers to a Value (calculated or not)

How to represent computational relationship between Blobs and Trees?
Encode → Value
Value → Blob, Tree
Blob, Tree → Thunk
Thunk → Encode
Fibonacci Sequence

```c
int fib ( int n ) {
    if ( n == 0 ) return 0;
    if ( n == 1 ) return 1;
    return ( fib (n - 1) + fib (n - 2) );
}
```
\((\text{fib} (n - 1) + \text{fib} (n - 2))\);
Fibonacci Sequence

```c
int fib ( int n ) {
    if ( n == 0 ) return 0;
    if ( n == 1 ) return 1;
    return ( fib (n-1) + fib (n-2) );
}
```

Evaluate:

- If `n == 0`, output 0
- If `n == 1`, output 1
- Output the sum of `fib(n-1)` and `fib(n-2)`
Representing deterministic computations with WebAssembly

```c
int fib ( int n ) {
    if ( n == 0 )  return 0;
    if ( n == 1 )  return 1;
    return ( fib (n - 1) + fib (n - 2) );
}
```

- Purely deterministic
- Guarantee memory safety
- Wasm is an LLVM target
- Low latency to invoke (<50ns) and high throughput (>85%)
Outline

- Introduction
- Our proposed solution: Fixpoint
- How is Fixpoint implemented
- Demo
- Open Questions
- Conclusion
Outline

• Introduction
• Our proposed solution: Fixpoint
• How is Fixpoint implemented
• Demo
• Open Questions
• Conclusion
Open Questions

• How general can Fixpoint be?
  ▪ Implementing WASI using Fixpoint API
  ▪ Can all programs be represented as a separation of deterministic computation and pre-declared dependencies?

• How fast can Fixpoint be?
  ▪ Performance of WebAssembly in Fixpoint
  ▪ Utilizing multi-memories for zero-copy access of Blobs and Trees
Outline

• Introduction
• Our proposed solution: Fixpoint
• How is Fixpoint implemented
• Demo
• Open Questions
• Conclusion
Takeaway

Separating processes into deterministic computations and pre-declared dependencies helps to solve these issues:

• Parable 1: The Sloppy Data Scientist
  - Lineage of computations that generate certain outputs is saved by Fixpoint
• Parable 2: The Anxious Cloud Storage Company
  - Computations are guaranteed to be deterministic
• Parable 3: The Inefficient Serverless Platform
  - Expose dependencies for making better placement decisions