SpiderLightning

A New Kind of Cloud Systems Interface

Jiaxiao (Joe) Zhou

Cloud_Native Rejekts [NA'22]

Jiaxiao Zhou

Twitter: @jiaxiao_zhou

GitHub: @Mossaka

- I’m an open source software engineer at @ DeisLabs Microsoft.
- I co-organize Systems Gossip Meetup - a virtual meetup dedicated to learn systems and programming languages.
- I love playing video games, hiking, and recently started weight lifting.

Cloud_Native Rejekts [NA'22]

Agenda

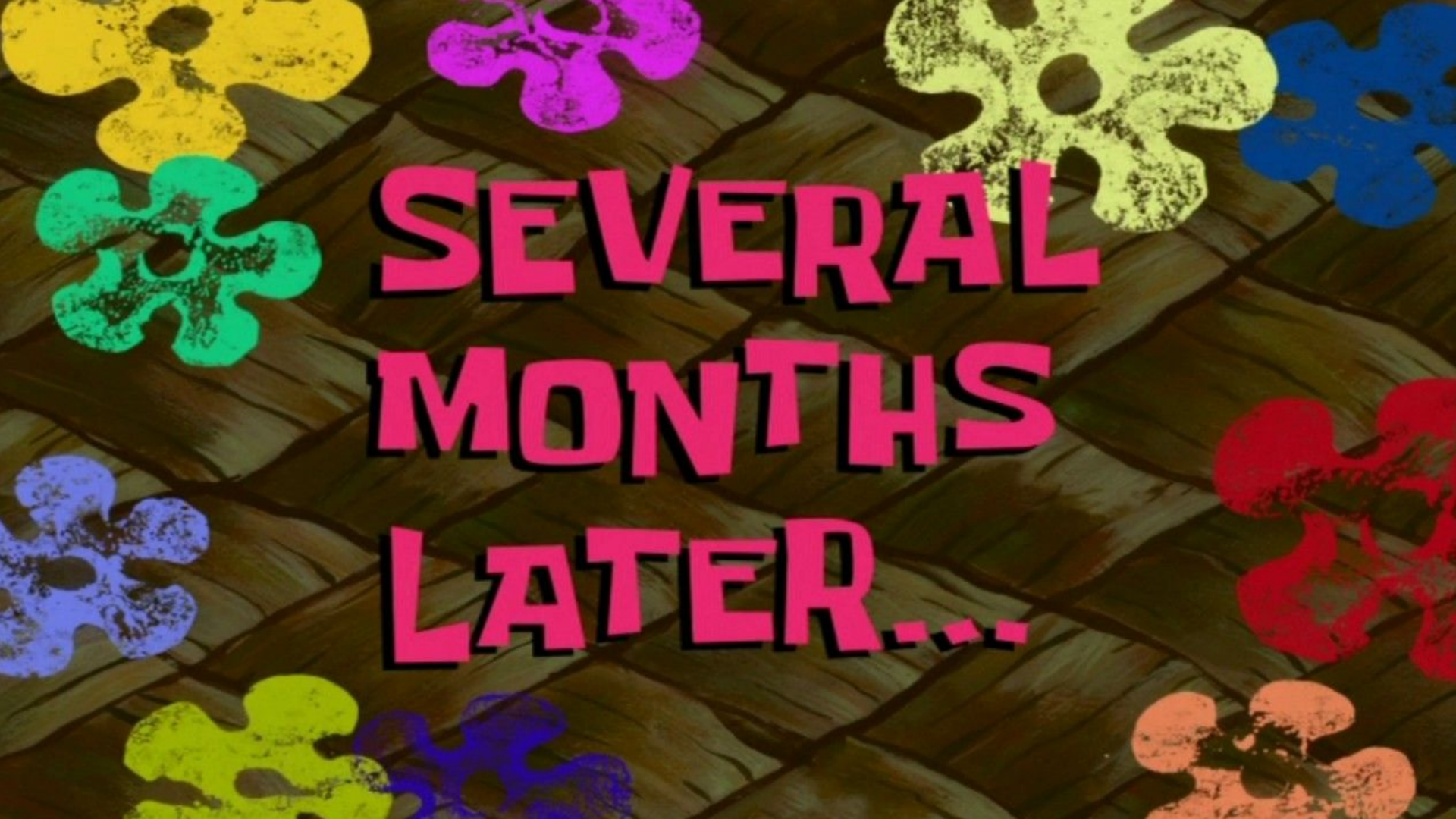
- Problem – Engineering Portable Distributed Apps is Hard…
- One Approach – SpiderLightning: A New kind of Cloud Systems Interface!
- A Deep Dive Into SpiderLightning, and… `slight`!
- Showcase: How far you can go with SpiderLightning and `slight`?
- A Call To Action.

Engineering Portable Distributed Apps is Hard…

Person A

I want you to build an on-premises service to do this, this, and that!

Company A


SEVERAL
MONTHS
LATER...

Actually, scratch that -
We want it on the Cloud!

Company A

... and on IoT devices, CDNs, and an Edge Network.

Actually, scratch that - We want it on the Cloud!

Company A

I want to explore other opportunities!

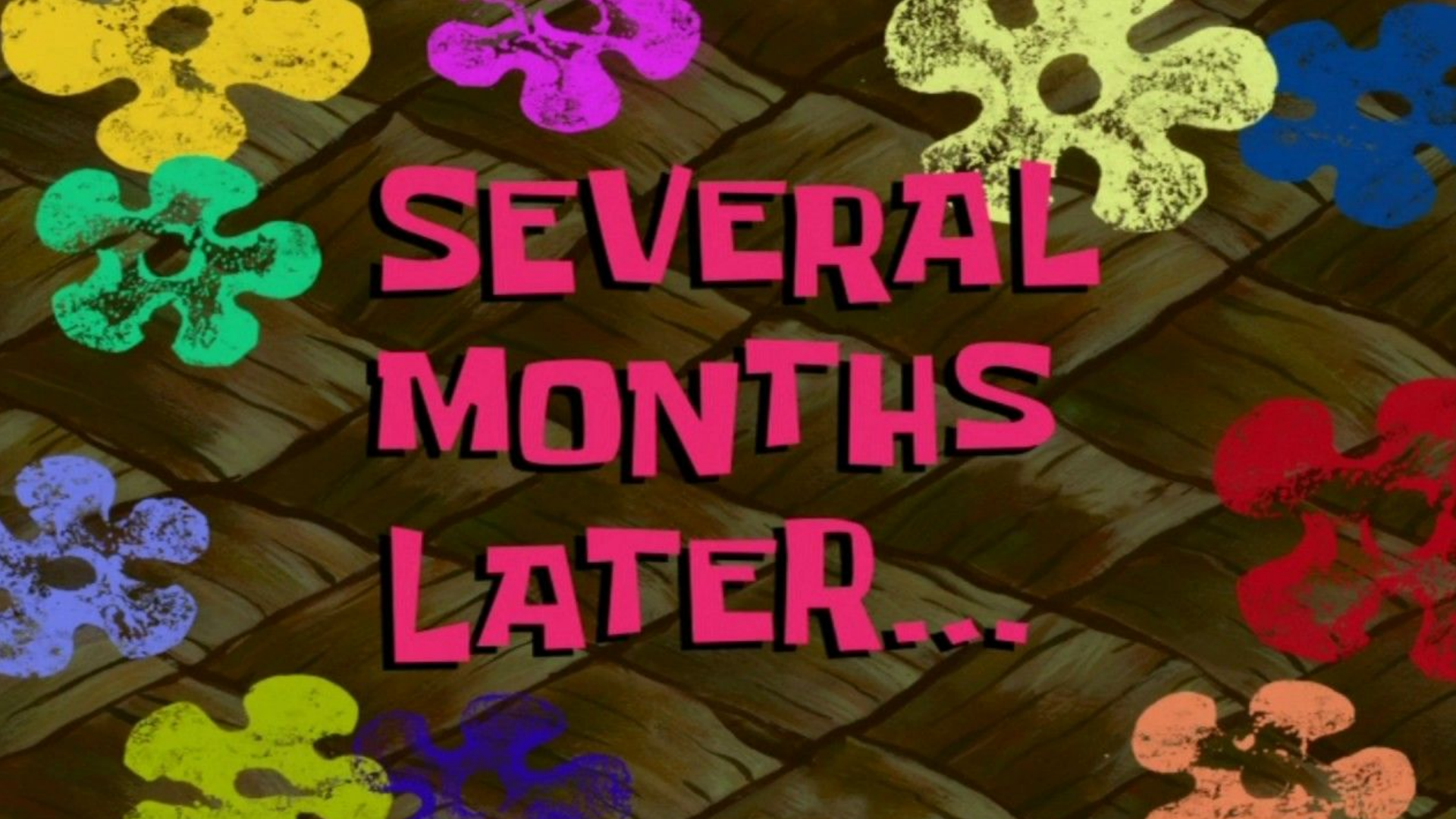
Person A

SpiderLightning
A New System Interface for the Cloud

Person A

I want you to build an on-premises service to do this, this, and that!

Company A


SEVERAL
MONTHS
LATER...

... and on IoT devices, CDNs, and an Edge Network.

Actually, scratch that - We want it on the Cloud!

Company A

ONE

LINE

CHANGE

How did SpiderLightning do all that?

lightweight + portable

from on-premises to cloud A with zero code changes

What is SpiderLightning?

What is SpiderLightning?

Message Queue

What is SpiderLightning?

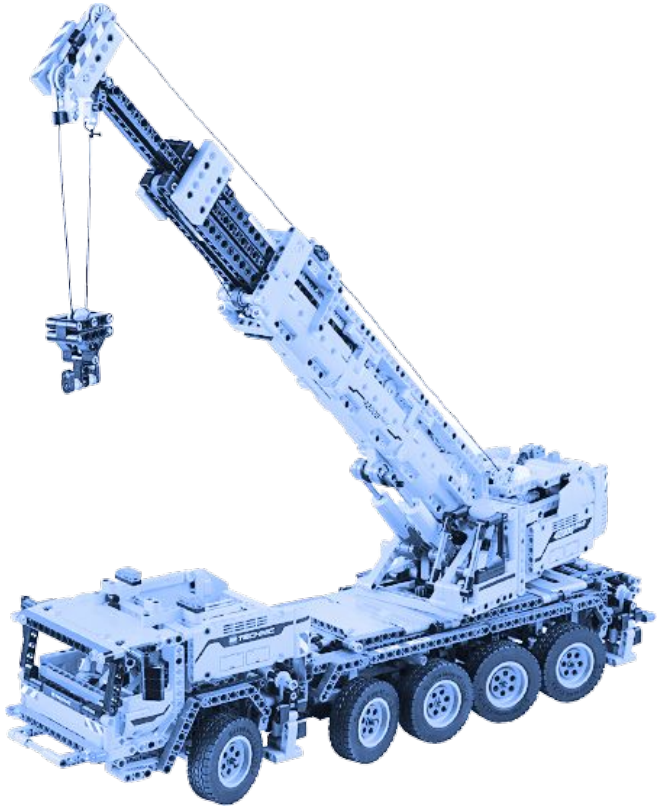
What is SpiderLightning?

What is SpiderLightning?

Cloud_Native Rejekts [NA'22]

What is SpiderLightning?

Mobile Crane MK II



Cloud_Native
Rejekts [NA'22]

The `slight` CLI

Cloud_Native
Rejekts [NA'22]

How can you get started with `slight` + SpiderLightning?

What? WebAssembly?

- A compilation target (binary instruction format) that
- brings C, C++, Rust and other languages to the browser and
- achieves near-native code performance
- in a compact form (compared to JavaScript).
- Runs in a virtual machine,
- has linear memory
- software sandboxing

Example

```
function fac() {

if n <= 1:

    return 1

else:

    return n * fac(n-1)

}
```

```
(module
    (func $fac (param $n i64) (result i64)
        ;; implementation
        local.get $n
        i64.const 1
        i64.le_s
        (if (result i64)
            (then (i64.const 1))
            (else
                (i64.sub (local.get $n) (i64.const
1))            call $fac
                local.get $n
                i64.mul
            )
        )
    )
    (export "fac" (func $fac))
)
```


Wasm + JavaScript!

```
WebAssembly.instantiateStreaming(fetch("fac.wasm"), importObject).then(
    (obj) => {
      // Call an exported function:
      obj.instance.exports.fac();
    }
);
```


WebAssembly Component Model

“A Wasm component is a deployable unit of software.” - Dan Gohman

- It is usable out of the box.
- It provides a way to define higher level types such as strings, records and variants.
- Multiple components can be dynamically linked together to form a bigger program.

WebAssembly Outside of the Browser

- **Security**: WebAssembly runtime inserts runtime checks that restrict the code to its own region of memory
- **Portable**: WebAssembly designed to be executable in various operating systems and architectures
- **Performance**: fast start-up time, and near-native performance

- Suitable for isolating code in clouds, edge networks, embedded devices, browsers, and network proxies.

WebAssembly System Interface (WASI)

- A standardization effort by WebAssembly CG of the W3C on
- a system interface API for WebAssembly to
- implement standard libraries like libc and portable across different operating systems
- A “virtual operating system” for a “virtual architecture”

```c
#include <stdio.h>
int main() {
  printf("Hello, World!");
  return 0;
}
```

```
main.wasm:      file format wasm 0x1

Section Details:

Import[5]:
 - func[0] sig=2 <__imported_wasi_snapshot_preview1_fd_close> <- wasi_snapshot_preview1.fd_close
 - func[1] sig=3 <__imported_wasi_snapshot_preview1_fd_fdstat_get> <- wasi_snapshot_preview1.fd_fdstat_get
 - func[2] sig=4 <__imported_wasi_snapshot_preview1_fd_seek> <- wasi_snapshot_preview1.fd_seek
 - func[3] sig=5 <__imported_wasi_snapshot_preview1_fd_write> <- wasi_snapshot_preview1.fd_write
 - func[4] sig=6 <__imported_wasi_snapshot_preview1_proc_exit> <- wasi_snapshot_preview1.proc_exit
```


SpiderLightning: A Cloud Systems Interface

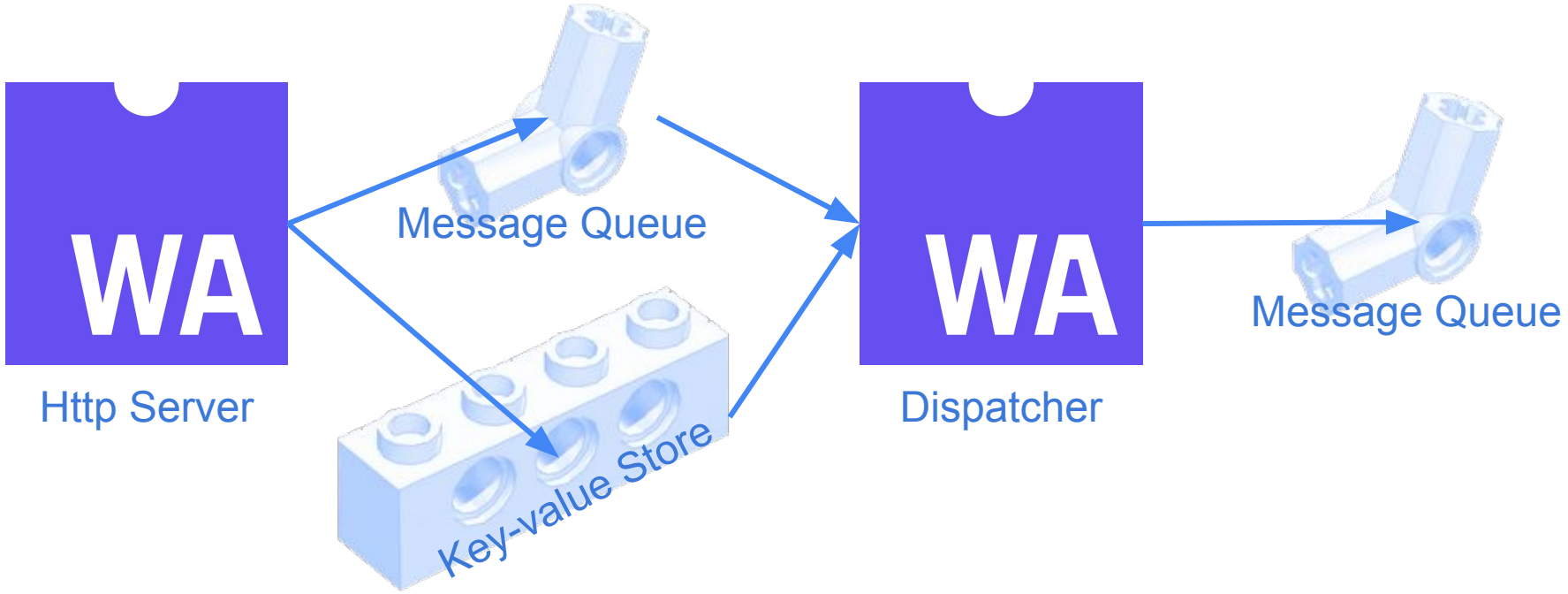
Cloud_Native Rejekts [NA'22]

WIT - A IDL for WebAssembly Components

```
// A key-value store interface.
use { error, payload } from types
use { observable } from resources

resource kv {
    // open a key-value store
    static open: func(name: string) -> expected<kv, error>

    // get the payload for a given key.
    get: func(key: string) -> expected<payload, error>

    // set the payload for a given key.
    set: func(key: string, value: payload) -> expected<unit, error>

    // list the keys in the store.
    keys: func() -> expected<list<string>, error>

    // delete the payload for a given key.
    delete: func(key:string) -> expected<unit, error>

    // watch for changes to a key.
    watch: func(key: string) -> expected<observable, error>
}
```

How far can you go with `slight` + SpiderLightning?

The Architecture Behind the Chat App



Let's build cool stuff together!

SpiderLightning GitHub Repository: https://github.com/deislabs/spiderlightning

Joe's Twitter: @jiaxiao_zhou