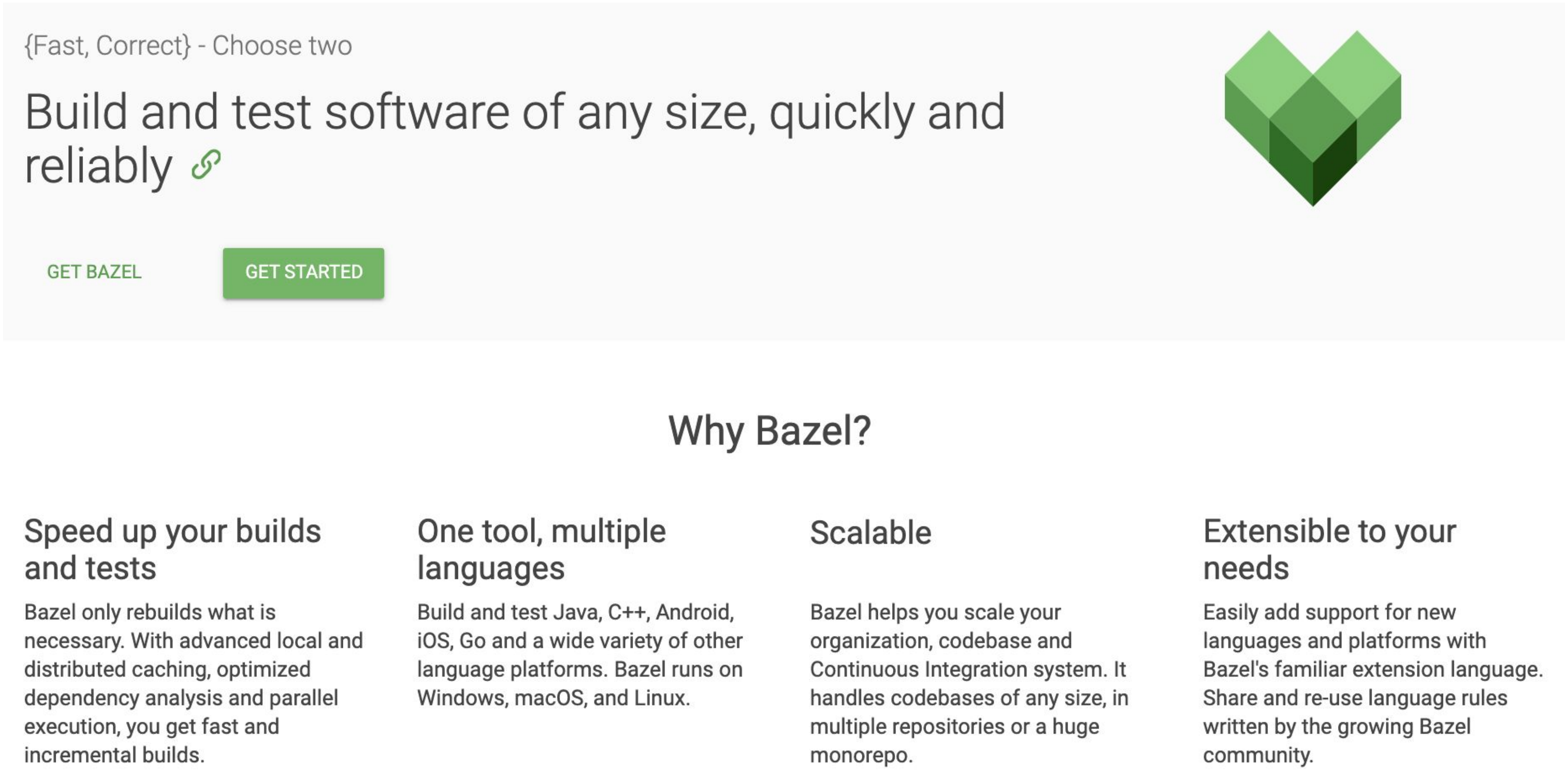
Optimizing Bazel Sandboxing with a FUSE File System

Overview of Bazel Sandboxing and sandboxfs
By Julio Merino (@jmmv) for FOSDEM on 2020-02-01

What is Bazel?



{Fast, Correct} - Choose two

Build and test software of any size, quickly and reliably

Why Bazel?

Speed up your builds and tests

Bazel only rebuilds what is necessary. With advanced local and distributed caching, optimized dependency analysis and parallel execution, you get fast and incremental builds.

One tool, multiple languages

Build and test Java, C++, Android, iOS, Go and a wide variety of other language platforms. Bazel runs on Windows, macOS, and Linux.

Scalable

Bazel helps you scale your organization, codebase and Continuous Integration system. It handles codebases of any size, in multiple repositories or a huge monorepo.

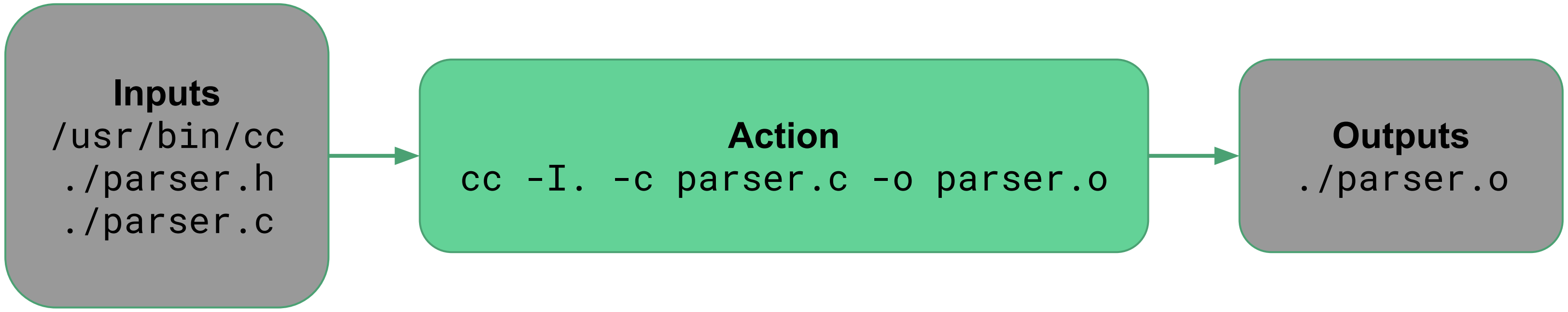
Extensible to your needs

Easily add support for new languages and platforms with Bazel's familiar extension language. Share and re-use language rules written by the growing Bazel community.

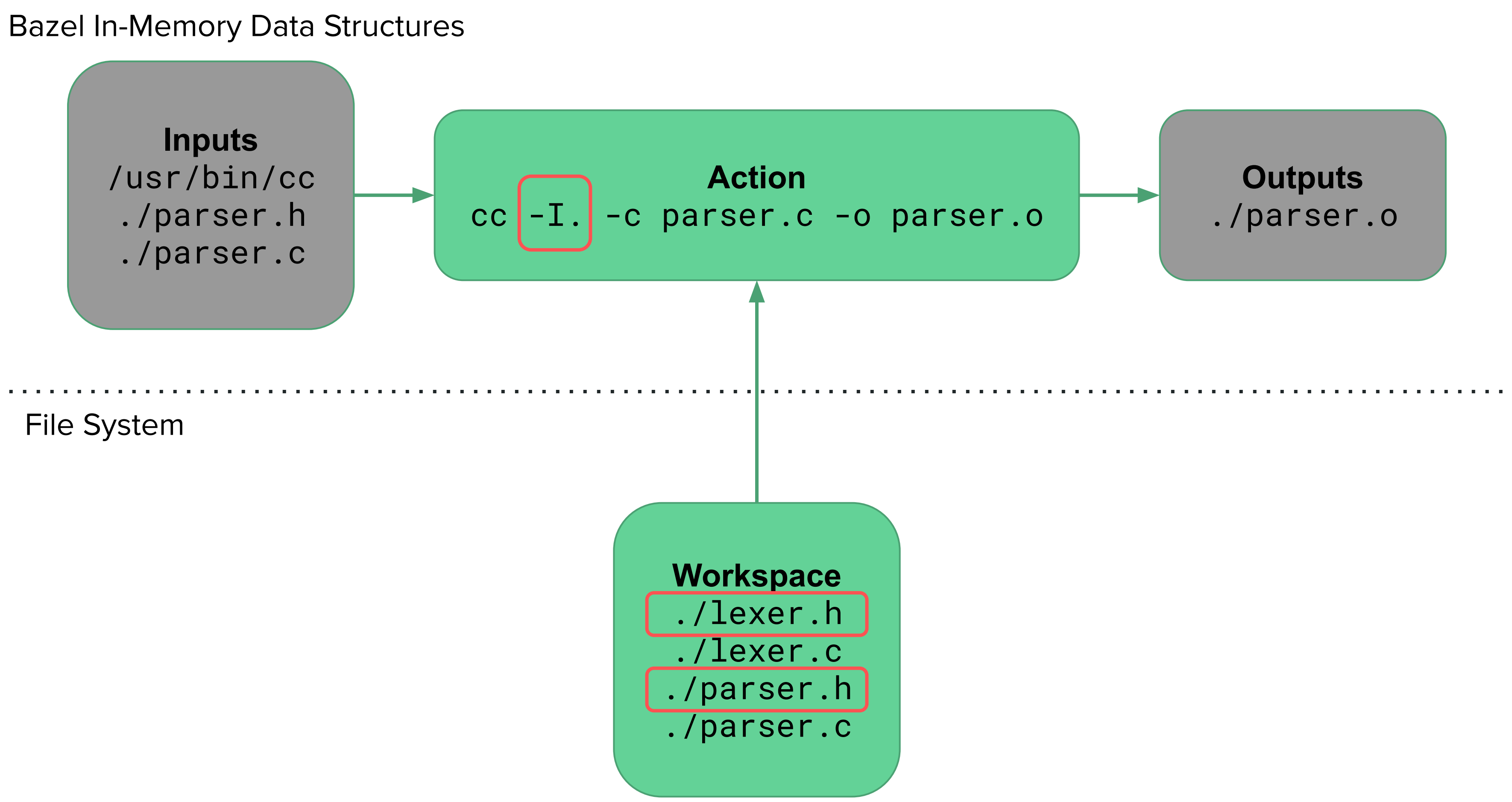
https://bazel.build/

What is an Action in Bazel?

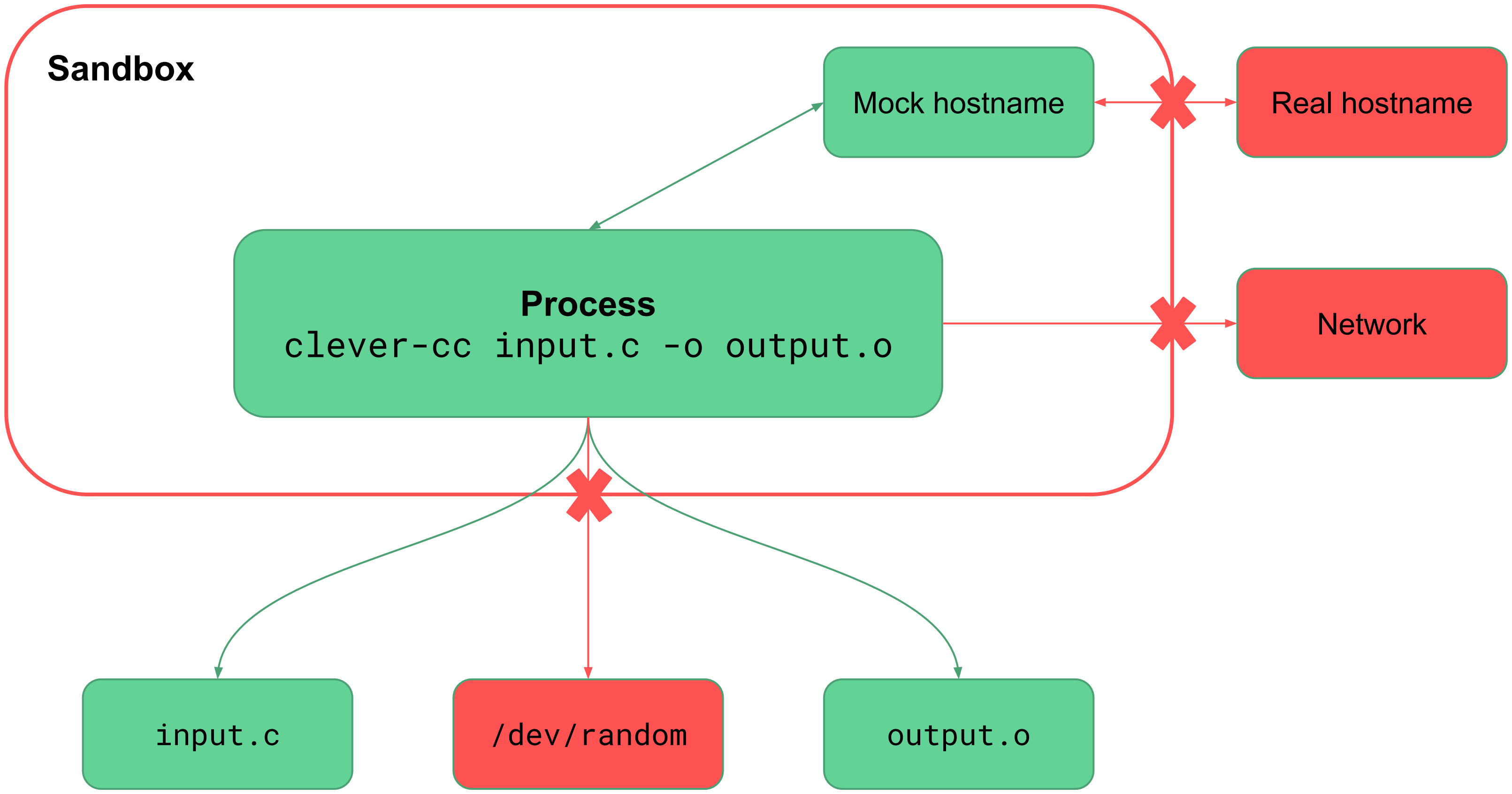
Bazel In-Memory Data Structures



Why Do We Sandbox Actions?

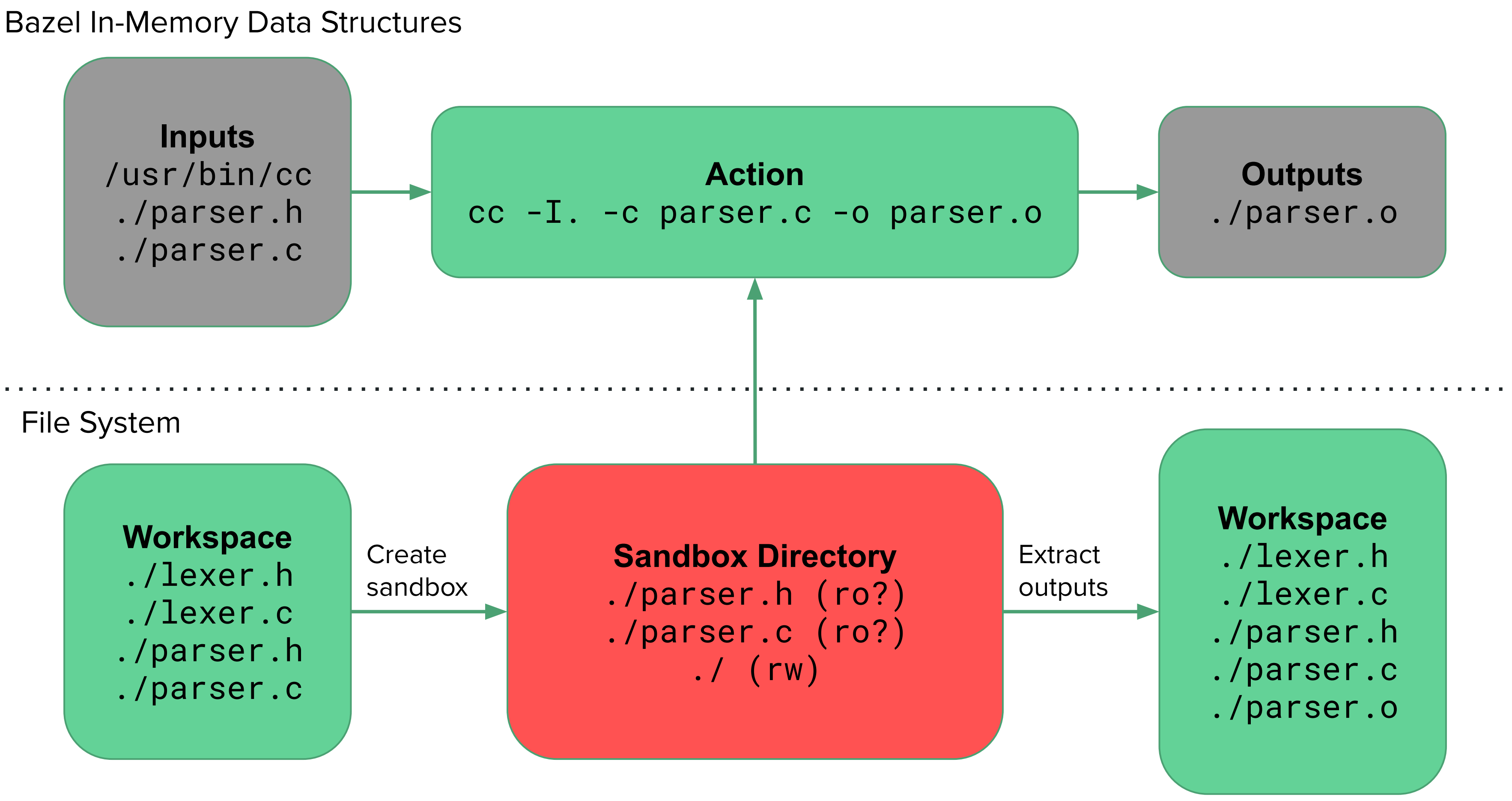


Action Sandboxing: Process Isolation

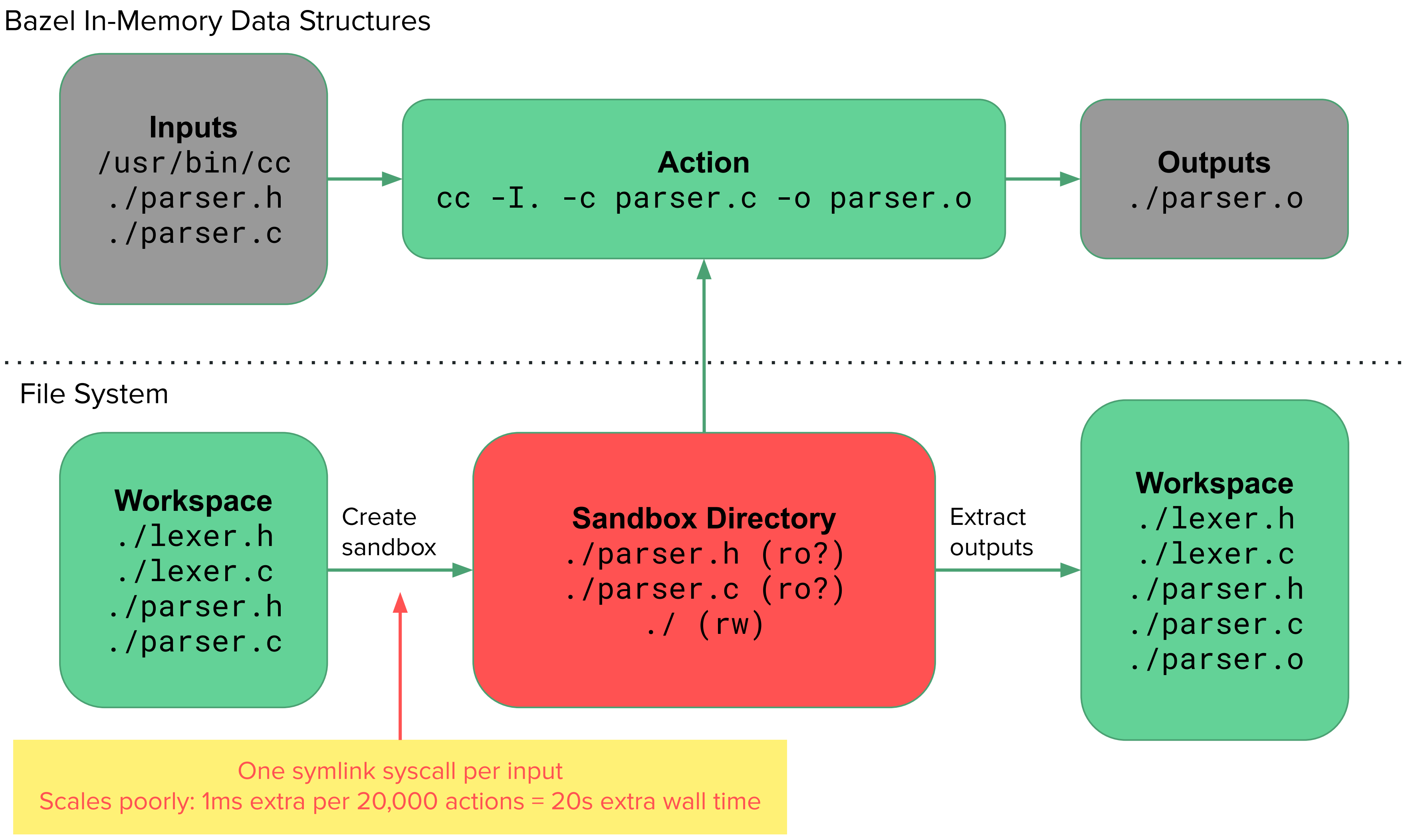


On Linux: user namespaces – On macOS: sandbox-exec
https://blog.bazel.build/2015/09/11/sandboxing.html, https://jmmv.dev/2019/11/macos-sandbox-exec.html

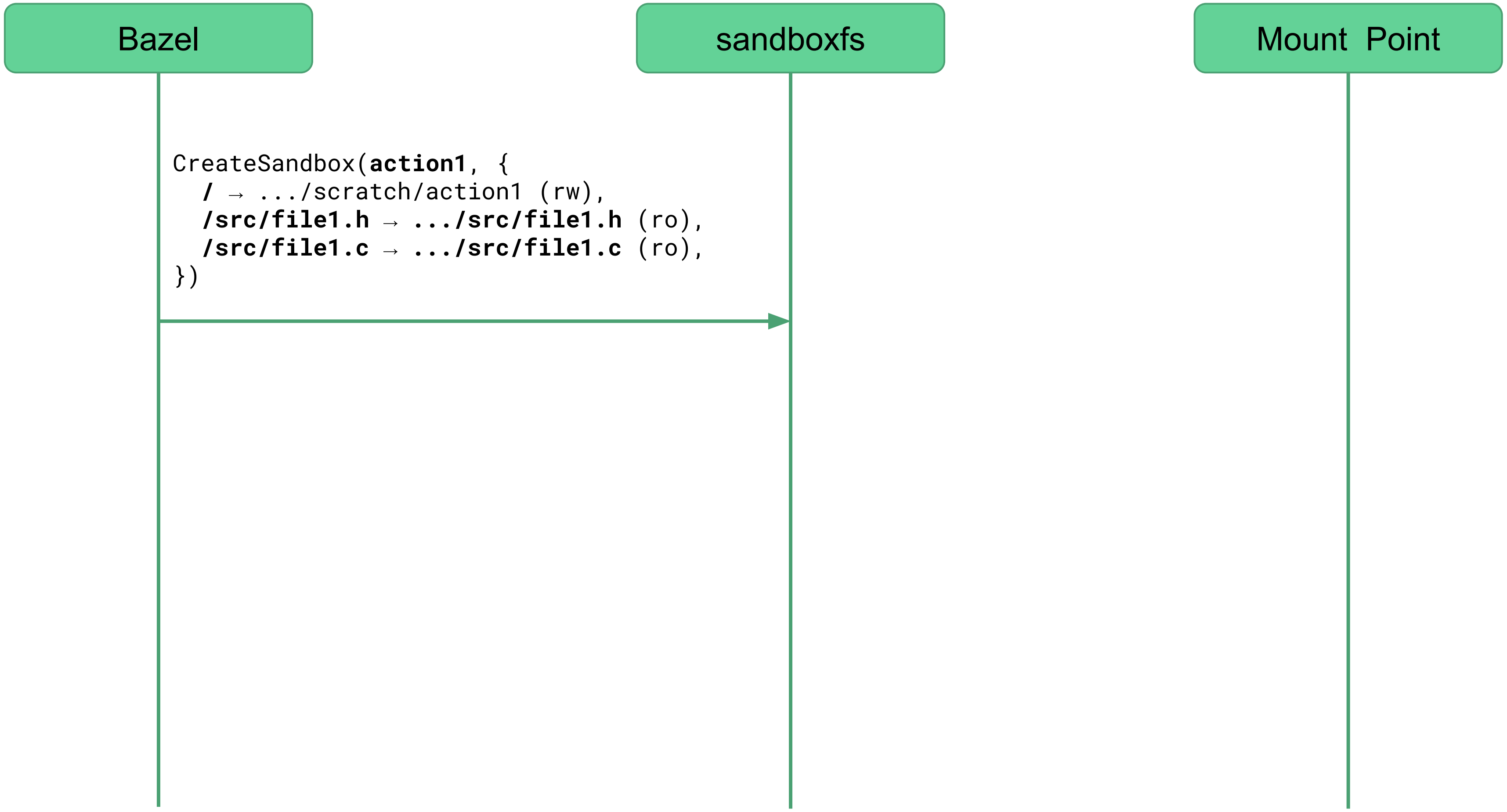
Action Sandboxing: File System Preparation



Action Sandboxing: File System Preparation



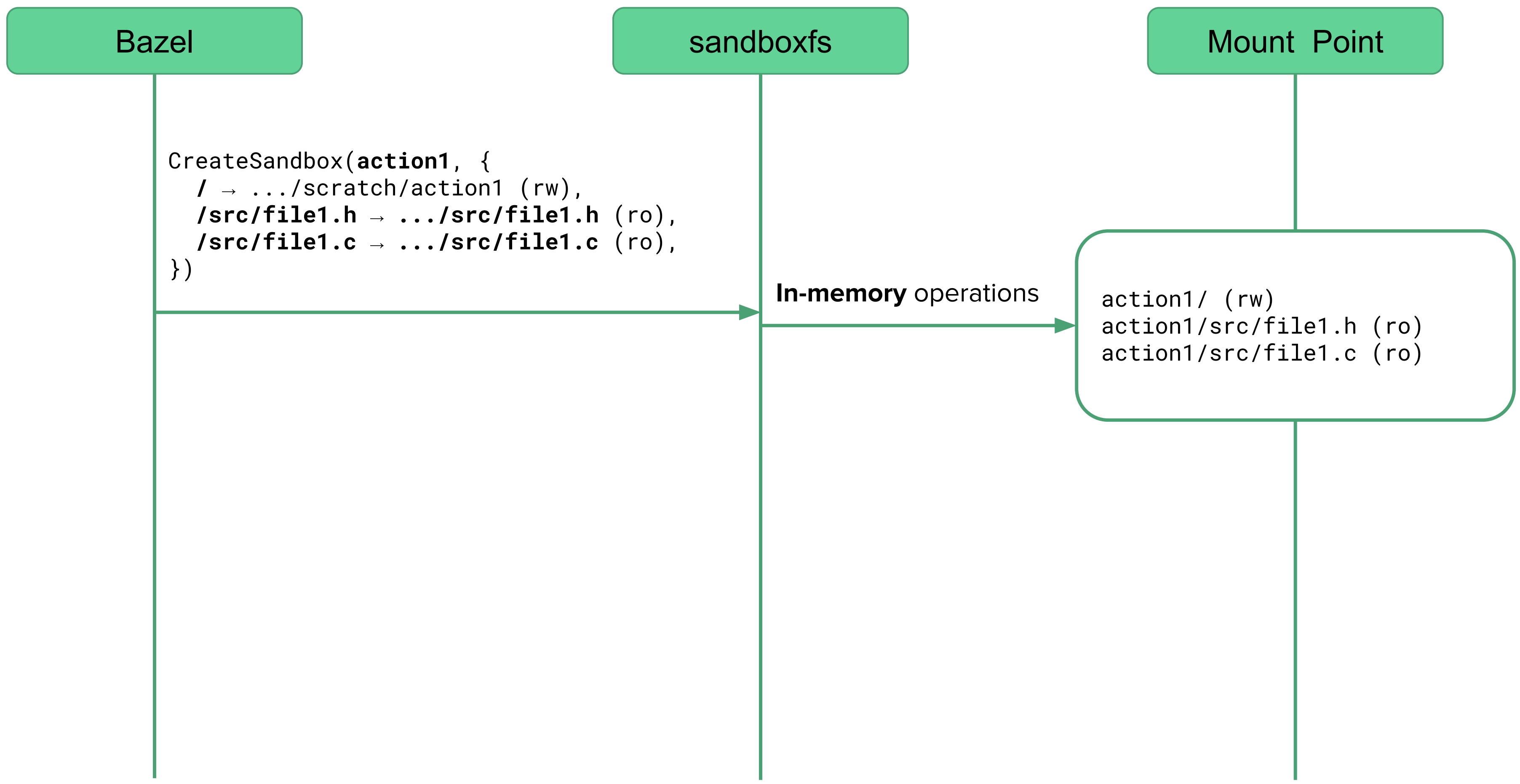
sandboxfs: Using FUSE to Avoid Symlinks



```
CreateSandbox(action1, {
  / → .../scratch/action1 (rw),
  /src/file1.h → .../src/file1.h (ro),
  /src/file1.c → .../src/file1.c (ro),
})
```

https://github.com/bazelbuild/sandboxfs/

sandboxfs: Using FUSE to Avoid Symlinks

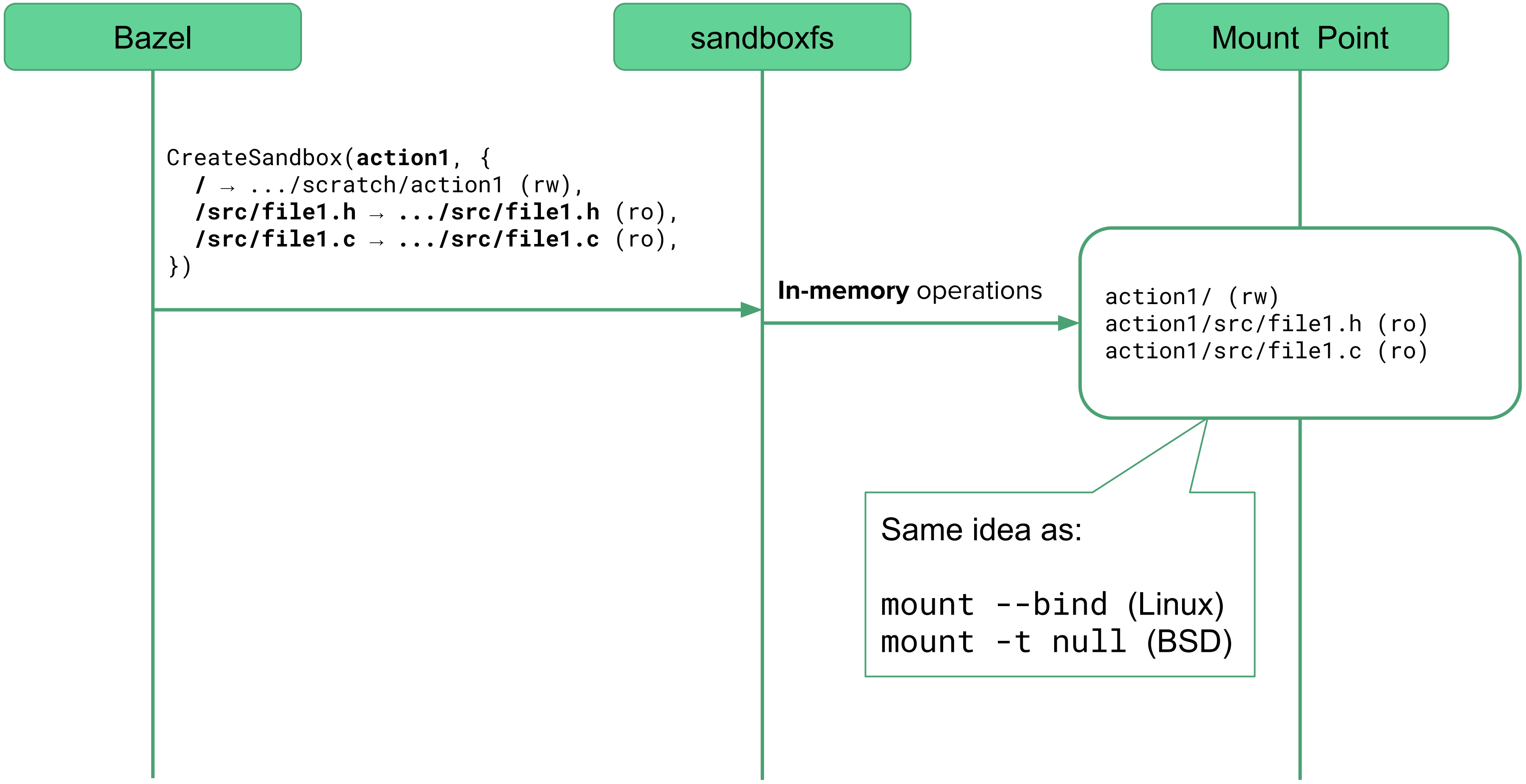


```
CreateSandbox(action1, {
  / → .../scratch/action1 (rw),
  /src/file1.h → .../src/file1.h (ro),
  /src/file1.c → .../src/file1.c (ro),
})
```

```
action1/ (rw)
action1/src/file1.h (ro)
action1/src/file1.c (ro)
```

https://github.com/bazelbuild/sandboxfs/

sandboxfs: Using FUSE to Avoid Symlinks



```
CreateSandbox(action1, {
  / → .../scratch/action1 (rw),
  /src/file1.h → .../src/file1.h (ro),
  /src/file1.c → .../src/file1.c (ro),
})
```

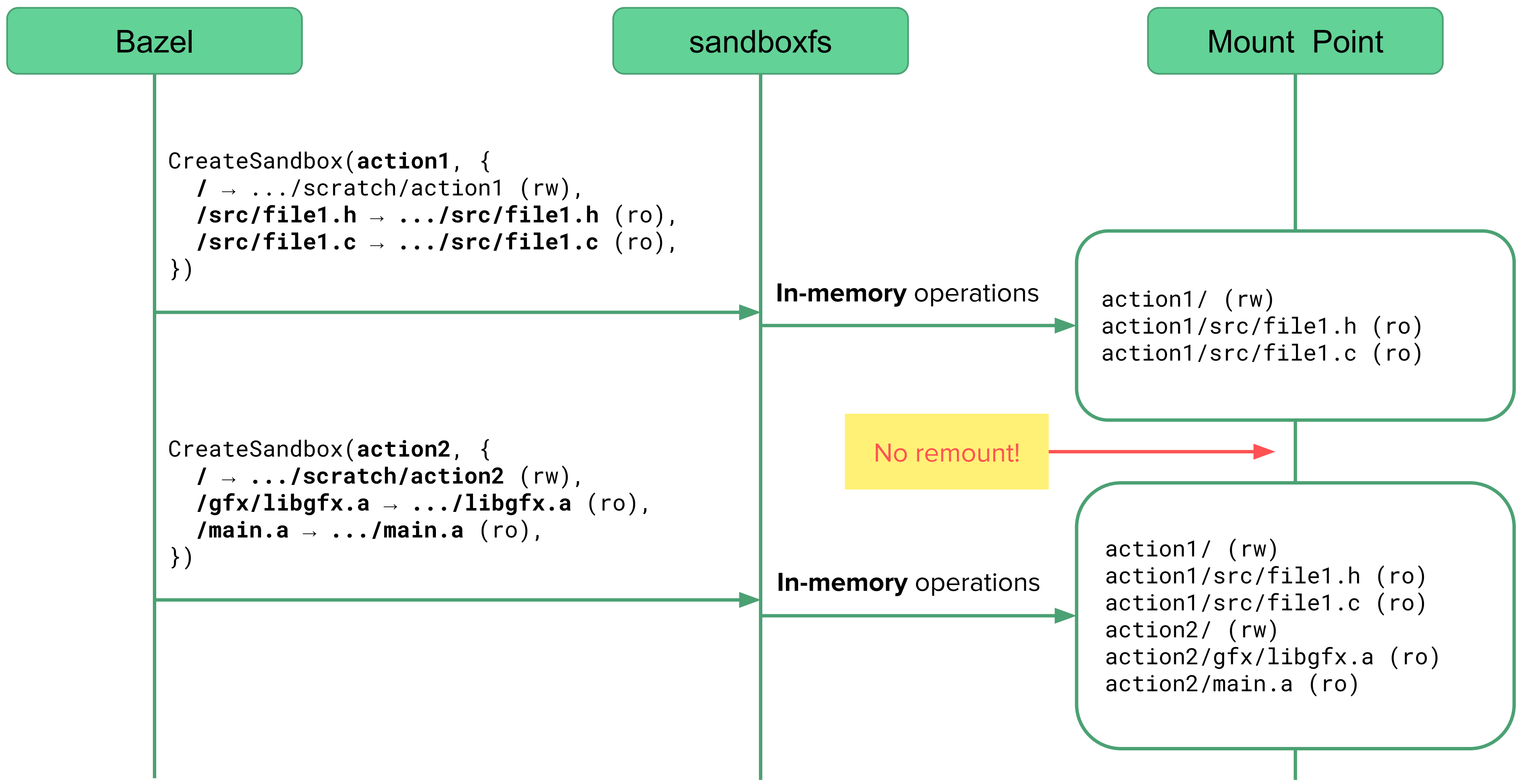
```
action1/ (rw)
action1/src/file1.h (ro)
action1/src/file1.c (ro)
```

Same idea as:

`mount --bind` (Linux)
`mount -t null` (BSD)

https://github.com/bazelbuild/sandboxfs/

sandboxfs: Using FUSE to Avoid Symlinks



```
CreateSandbox(action1, {
  / → .../scratch/action1 (rw),
  /src/file1.h → .../src/file1.h (ro),
  /src/file1.c → .../src/file1.c (ro),
})
```

```
action1/ (rw)
action1/src/file1.h (ro)
action1/src/file1.c (ro)
```

```
CreateSandbox(action2, {
  / → .../scratch/action2 (rw),
  /gfx/libgfx.a → .../libgfx.a (ro),
  /main.a → .../main.a (ro),
})
```

```
action1/ (rw)
action1/src/file1.h (ro)
action1/src/file1.c (ro)
action2/ (rw)
action2/gfx/libgfx.a (ro)
action2/main.a (ro)
```

https://github.com/bazelbuild/sandboxfs/

Performance Results (macOS, circa 2018)

Target	Machine	Local
Bazel	MacBook Pro 2017	581s
Bazel	Mac Pro 2013	247s
iOS app	Mac Pro 2013	1235s

https://blog.bazel.build/2018/04/13/preliminary-sandboxfs-support.html

Performance Results (macOS, circa 2018)

Target	Machine	Local	Symlinked sandbox
Bazel	MacBook Pro 2017	581s	621s (+6%)
Bazel	Mac Pro 2013	247s	265s (+7%)
iOS app	Mac Pro 2013	1235s	4572s (+270%)

https://blog.bazel.build/2018/04/13/preliminary-sandboxfs-support.html

Performance Results (macOS, circa 2018)

Target	Machine	Local	Symlinked sandbox	sandboxfs sandbox
Bazel	MacBook Pro 2017	581s	621s (+6%)	612s (+5%)
Bazel	Mac Pro 2013	247s	265s (+7%)	250s (+1%)
iOS app	Mac Pro 2013	1235s	4572s (+270%)	1922s (+55%)

https://blog.bazel.build/2018/04/13/preliminary-sandboxfs-support.html

Rewriting sandboxfs from Go to Rust

Go

- Easy to write (intern project!)
- VSCode has great support
- Hit scalability problems
- Started to become hard to maintain (few annotations in the code)

https://jmmv.dev/2018/07/rust-vs-go.html

Rewriting sandboxfs from Go to Rust

Go

- Easy to write (intern project!)
- VSCode has great support
- Hit scalability problems
- Started to become hard to maintain (few annotations in the code)

Rust

- Harder to write
- VSCode has support but is very slow
- Much more confident in the code
- In translating the old Go code, Rust uncovered many concurrency issues in it

https://jmmv.dev/2018/07/rust-vs-go.html

Rewriting sandboxfs from Go to Rust

Go

- Easy to write (intern project!)
- VSCode has great support
- Hit scalability problems
- Started to become hard to maintain (few annotations in the code)

Rust

- Harder to write
- VSCode has support but is very slow
- Much more confident in the code
- In translating the old Go code, Rust uncovered many concurrency issues in it

Commonalities

- pprof for finding performance issues
- FUSE bindings not first-class

https://jmmv.dev/2018/07/rust-vs-go.html

Future Work

- Optimize further
  - Current Bazel ↔ sandboxfs protocol very inefficient
- pkg_comp
- Other sandboxing approaches (Microsoft's BuildXL)
- But... beware of OSXFUSE and kexts on Mac

Thank You!

https://bazel.build/

https://github.com/bazelbuild/sandboxfs/

https://jmmv.dev/

https://twitter.com/jmmv/