



BazelCon 2026

Sandbox is a filesystem

Alternate sandboxing strategies for Bazel



Sahin Yort

Sr. Engineer at Aspect

#BazelCon

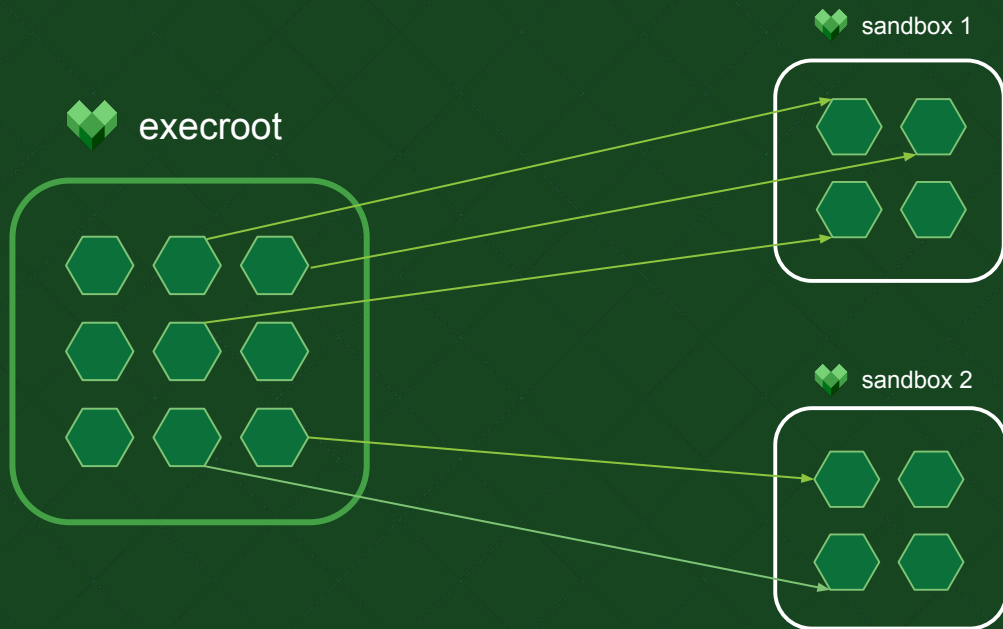


What is a Sandbox?

- * It is a sandwich box



How Bazel builds a sandbox on macOS



Horror story: macOS



tsc app.js -o app.dist.js

```
1 let path = fs.readlinkSync("app.js");  
2  
3 let dirname = path.dirname(path);  
4  
5 let entries = fs.readdirSync(dirname);  
6  
7 for (entry in entries) {  
8   if (entry.endsWith("app.d.ts")) {  
9     let contents = fs.readFileSync(declaration);  
10    //oops!  
11  }  
12 }
```

"/project/app.js" — the real path, leaked

"/project/" — absolute dirname

app.js, app.d.ts, secret.ts, every undeclared inputs

EPERM — Seatbelt denies it, the build fails 💥



BazelCon 2026

The rules_js patch stopped working



BazelCon 2026



BazelCon 2026

FSKit

Apple's userspace filesystem API

#BazelCon



FSKit ships as an AppEx



Problems with FSKit

- Mount cost — too slow to mount one per sandbox



Problems with FSKit

- Mount cost — too slow to mount one per sandbox
- Every cache miss is a kernel → userspace roundtrip



Problems with FSKit

- Mount cost — too slow to mount one per sandbox
- Every cache miss is a kernel → userspace roundtrip
- Transparency, Consent, and Control — permission requirement



Problems with FSKit

- Mount cost — too slow to mount one per sandbox
- Every cache miss is a kernel → userspace roundtrip
- Transparency, Consent, and Control — permission requirement
- Large inputs cost one user space → kernel round trip



Problems with FSKit

- Mount cost — too slow to mount one per sandbox
- Every cache miss is a kernel → userspace roundtrip
- Transparency, Consent, and Control — permission requirement
- Large inputs cost one user space → kernel round trip
- AMFI & Gatekeeper — every cold inode is scanned on first exec

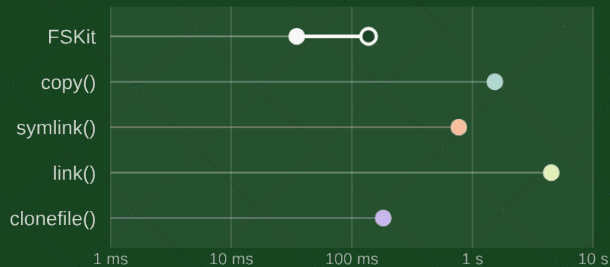


Cost to lay the tree down

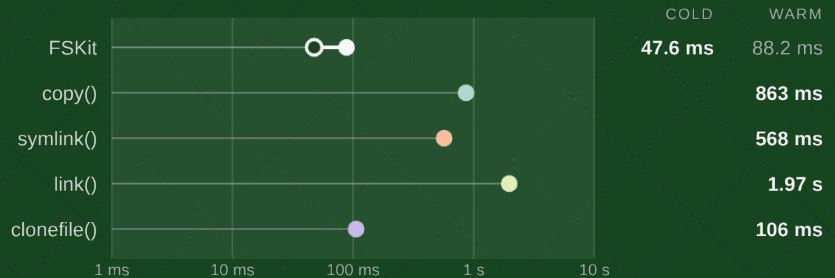
Time to make the files appear where the build expects them — log scale, four orders of magnitude apart. FSKit cold = first create on a fresh mount.

● FSKit ● copy() ● symlink() ● link() ● clonefile() ○ cold ● warm

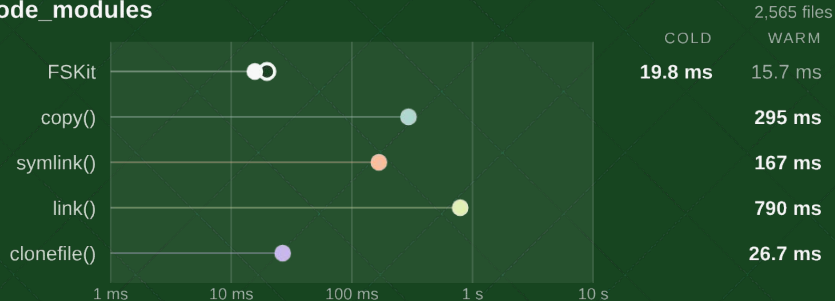
many small files



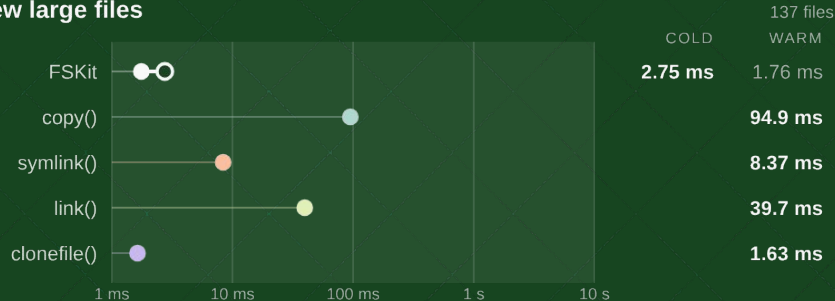
deep tree



node_modules



few large files



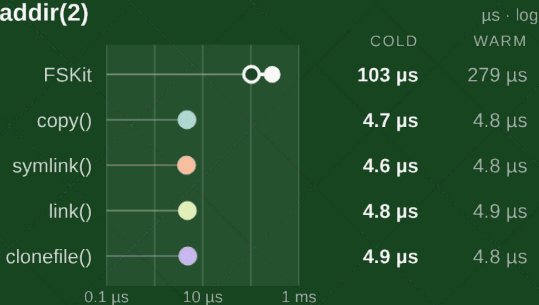
BazelCon 2026

What you pay back on every read

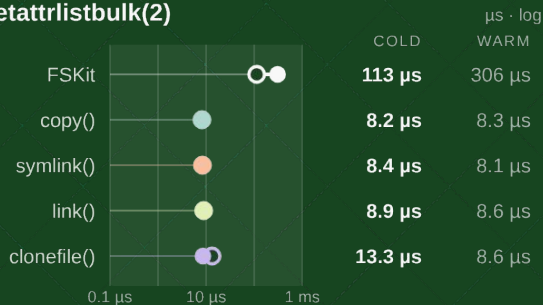
Per-call p50 on the many-small-files tree (16,093 files), median of 3 runs. Every uncached FSKit call is a kernel → userspace round trip: ~100 µs.

● FSKit ● copy() ● symlink() ● link() ● clonefile() ○ cold ● warm

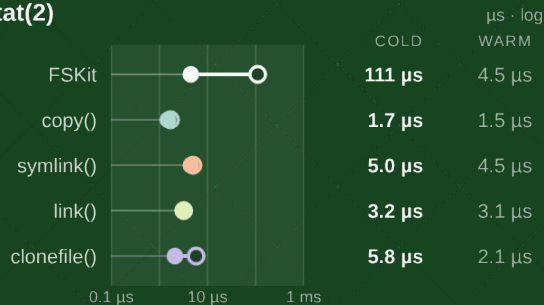
readdir(2)



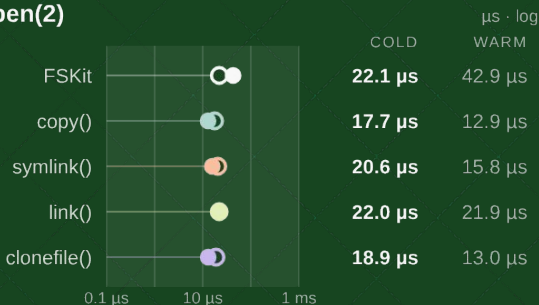
getattrlistbulk(2)



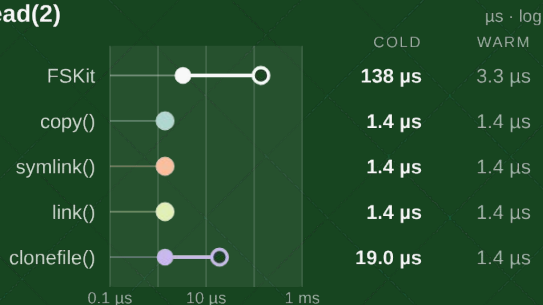
stat(2)



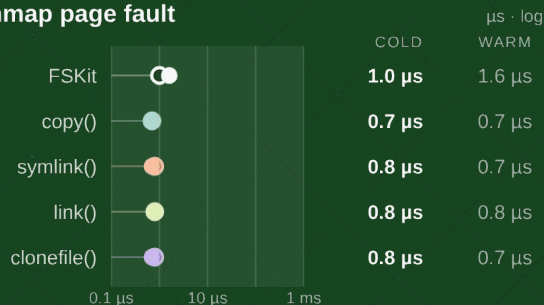
open(2)



read(2)



mmap page fault



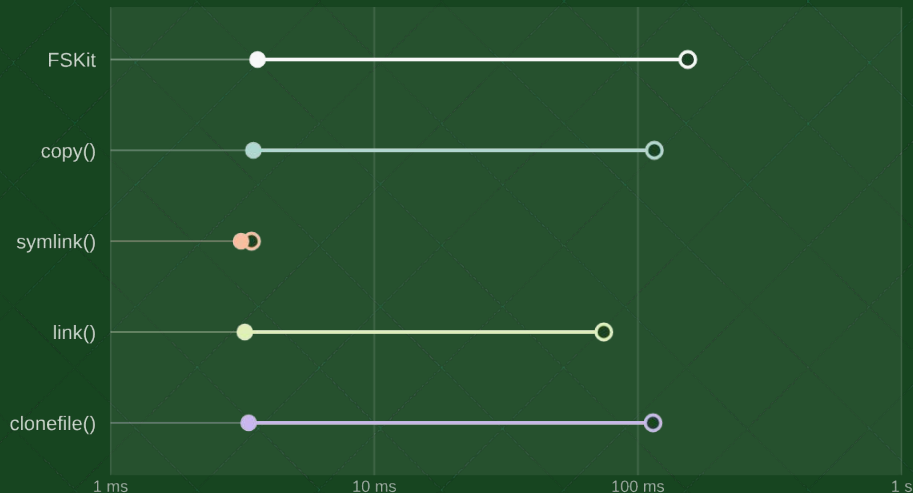
BazelCon 2026

Every new inode pays AMFI again

fork + exec + wait of a 16 MB binary already run once on the host, median across all four trees. Cold = first launch from this backend, warm = p50 of 30 launches.

● FSKit ● copy() ● symlink() ● link() ● clonefile() ○ cold ● warm

execve(2)



ms - log
COLD WARM

155 ms 3.61 ms

115 ms 3.48 ms

3.42 ms 3.13 ms

74.2 ms 3.23 ms

114 ms 3.34 ms

Shared inode

3.4 ms

symlink() reuses the host's already-validated inode.
Nothing to pay.

New inode

114–155 ms

AMFI validates every page again. FSKit is worst:
each page comes back out through userspace.

Also on the bill

107 ms

Mounting FSKit, once per workspace.



BazelCon 2026

Verdict

	● FSKit	● copy()	● symlink()	● link()	● clonefile()
Sandbox escape	None	None	readlink leaks the real path	Writes reach the source	None
Lay down 16k files	137 ms	1.53 s	769 ms	4.49 s	182 ms
First exec, 16 MB tool	155 ms	115 ms	3.4 ms	74.2 ms	114 ms
Walk + grep 16k files	938 ms	432 ms	633 ms	607 ms	457 ms
Also pays	107 ms mount · TCC consent	Every byte, every action	Undeclared inputs visible	Same volume only	APFS, same volume only

Many-small-files tree (16,093 files), median of 3 runs. First exec is the median across all four trees. Green = best in row.



BazelCon 2026

So we built it: SandboxFS

A pluggable sandbox protocol

```
common --sandbox_backend=my-sandbox=sandboxfs  
common --spawn_strategy=my-sandbox,local
```



We need a local execution protocol

Why not use Remote Execution protocol?



How does it work?



The sandbox was always a filesystem

macOS

FSKit



eBPF / NS /
Landlock



freeBSD

Jails



BazelCon 2026



Resources



[bazelbuild/bazel#29165](https://github.com/bazelbuild/bazel#29165)



[aspect-build/sandboxfs](https://github.com/aspect-build/sandboxfs)



BazelCon 2026

