Linux Gaming in 2020

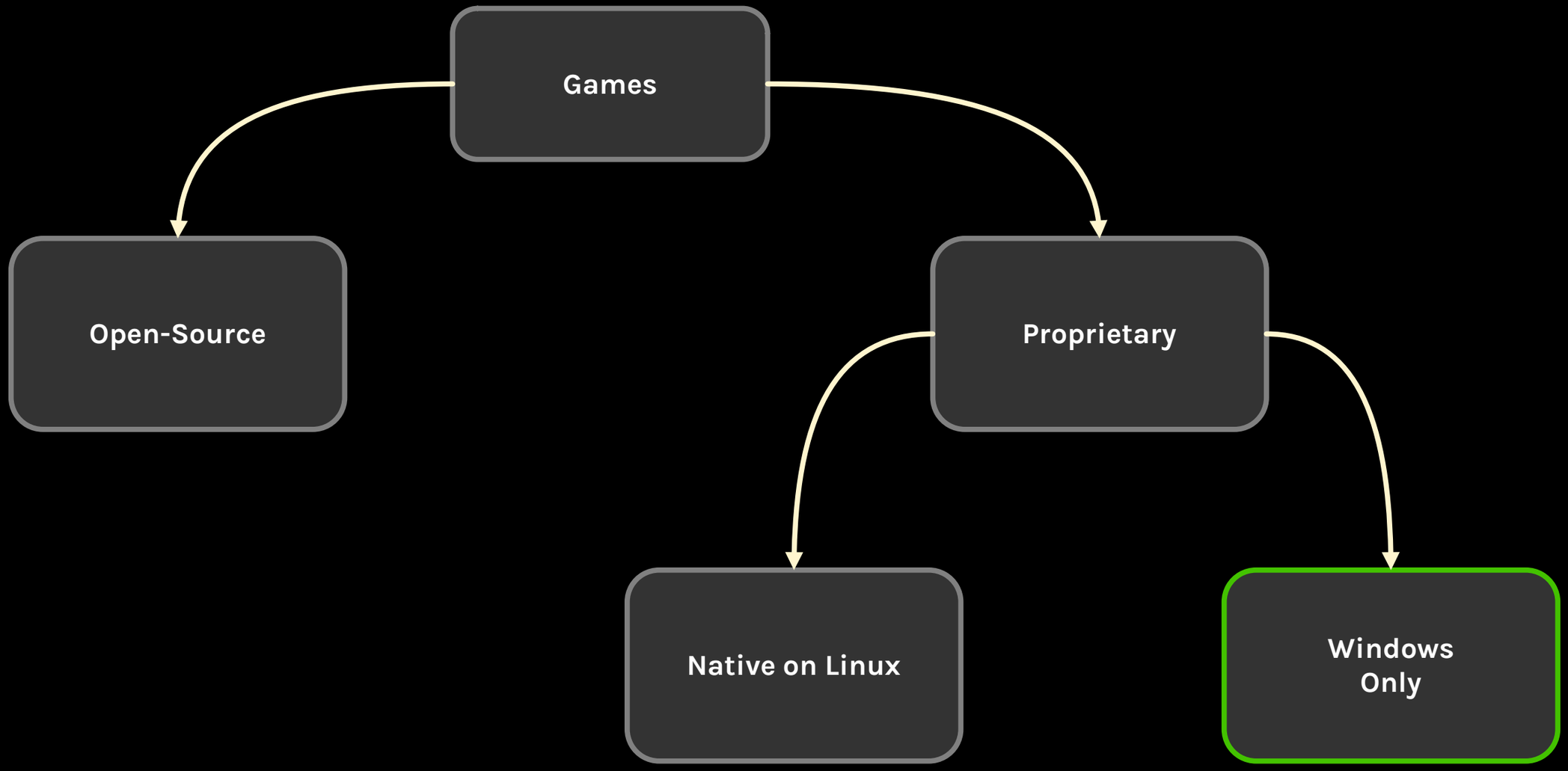
Preparing the Linux Kernel to emulate modern Windows Games

Gabriel Krisman Bertazi

krisman@collabora.com

Open First

Games on Linux



Vast Majority of Games

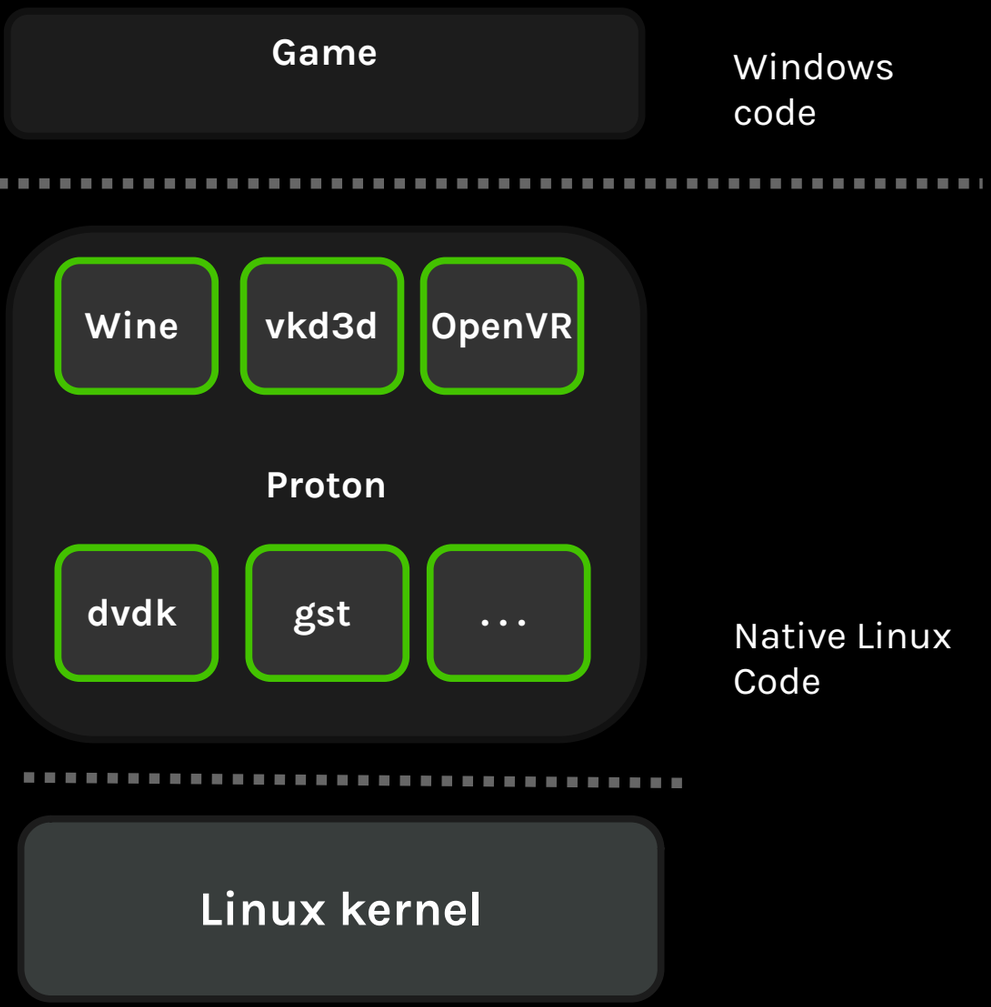
- Proprietary applications
- No support from game studios
- No recompilation possible
- DRM locks, anti-cheat mechanisms: Stack Protection
- Coded for Windows
- Linux traction: a Chicken-Egg problem
  - Games availability (Developers interest) X platform usage

Circumvent the problem

- Bring the game ecosystem to Linux through emulation
- Asbtract Windows Environment. A compatibility layer
- No recompiling
- Proton: Compatibility tool to run Windows Games on Linux
- Open-Source tool maintained by Valve
- Big Piece: Wine

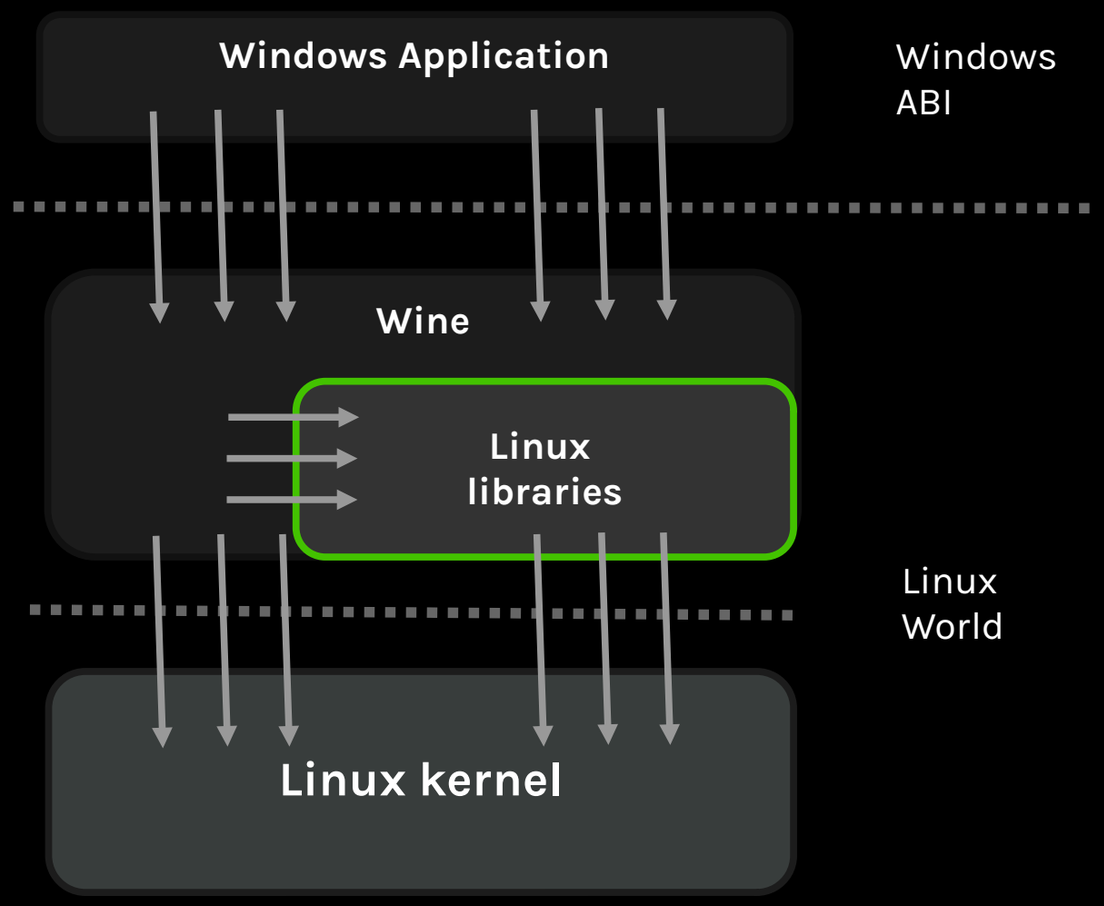
Proton

- Environment to run Windows games over Linux
- Bring together & configure stack to run game on Wine
- Used by the Steam Client to launch games



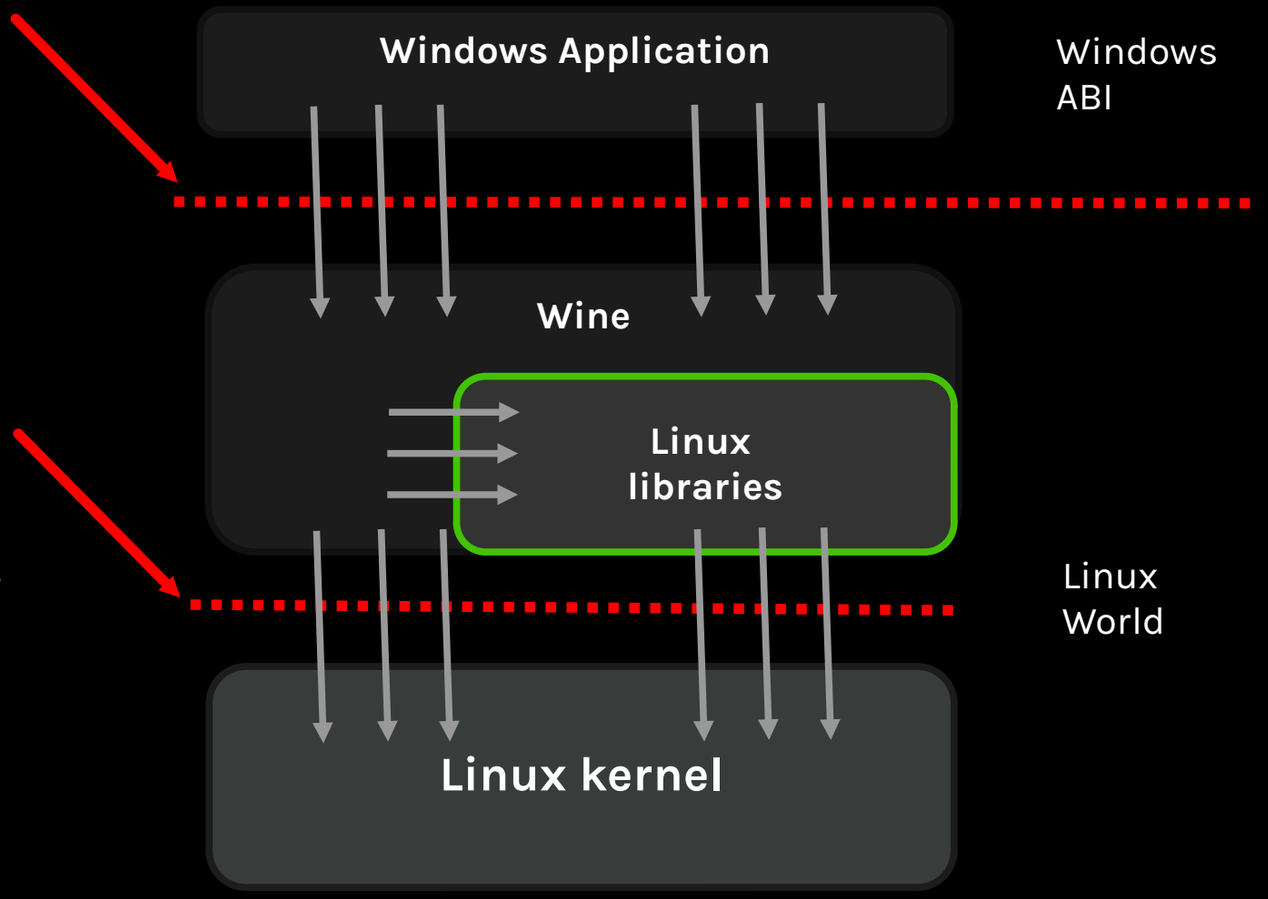
Wine: The big piece

- Compatibility layer
- Not a security sandbox
- No virtualization
- Implements WinAPI
- Exposes expected windows ecosystem to the application
- Community developed
- Major contributions by **Valve** and **CodeWeavers**



Very different ecossystems

- WinAPI
- Linux Programming interface (POSIX-like, with extensions)
- High burden on Wine
  - Hard to Emulate interfaces
  - Increased overhead



Our kernel effort

- Identify pain points for Wine emulation with current kernel
- Introduce new features in Linux to optimize Wine emulation
- It is not turning Linux into a copy of Windows
- It is not reimplementing Windows APIs in the kernel
- Only solve things that cannot be solved in userspace
  - Either due to performance or correcteness

Pain Point #1: Filesystems

Pain Point #1: Filesystems

- Historically, Windows exposes natively Case-Insensitive filesystems
  - But common Linux filesystems are all case-sensitive
- Case-Insensitive

In a case-insensitive filesystem, the following are the same file

```
/home/krisman/.games/file1
/home/krisman/.games/FILE1
/home/krisman/.games/FiLe1
```

Pain Point #1: Filesystems

- Games rely on this behavior.
- And will crash because of it
- Wine needs to expose that Case-insensitive file name lookup interface
- Proton doesn’t choose your filesystem
- Wine can emulate it in userspace at a high cost

Case-insensitive filesystems emulation

- We usually look for a userspace solution first
- Fully featured userspace implementation available
  - Inherent performance problems
  - Unsolveable race-conditions
  - Affecting real games functionalities
- What is the right solution?

Case-insensitive filesystems

- Native Case-insensitive support in selected Linux filesystems
- Solves the pain points
  - Performance
  - Correcteness
- Small backslash from development community
  - But turns out, other people needed it too
- Big backslash from user community
  - Don’t break my system!

Case-insensitive - FQA

- Supported since Linux 5.4
- It won’t make your system case-insensitive
  - It is per-directory
- It will never be a default setting
  - It is designed to not be configurable on the root filesystem
- It is for specific use cases
  - Wine and applications that need it can enable it on their internal directories
- It is safe
  - It only affects specific directories!
- But, it requires distros support
  - Build support for the feature: kernel support and instalation time configuration

Case-insensitive - Distros

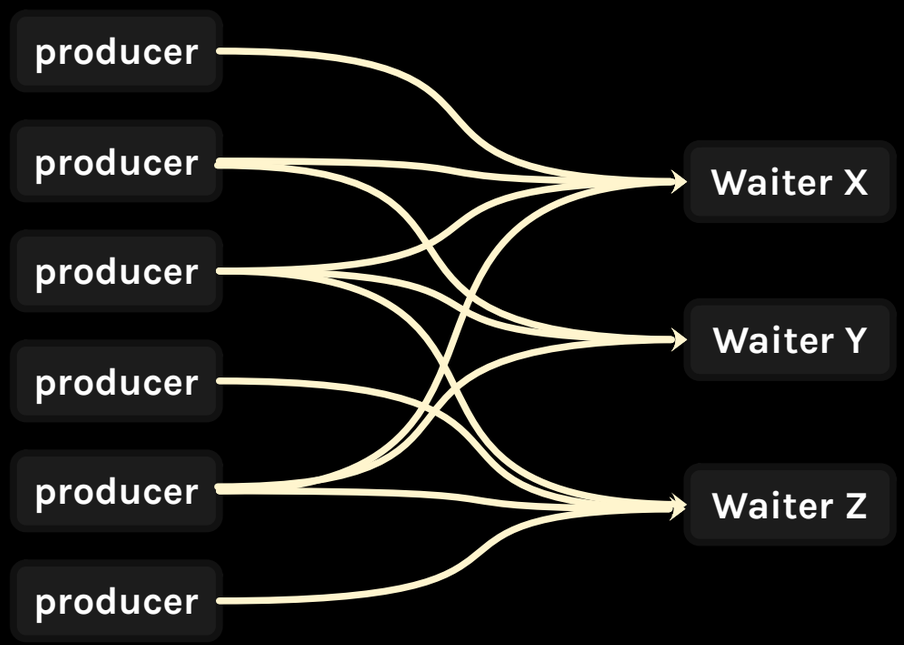
- In order for this feature to be usable
  - distro kernel must have `CONFIG_UNICODE`
  - Installation-time disk-wide configuration for Encoding support
  - Or, upcoming version of `e2fsprogs`
- Many distros still don’t enable `CONFIG_UNICODE`
- Many distros don’t enable `encoding` at installation
  - No problem in doing that, doesn’t change default semantics

Case-insensitive - What's next?

- Improving `fsck` support, to guarantee recovery of corrupted filesystems with that flag enabled
- Pushing for distros to adopt it
- Support more filesystems
  - Same semantics
- Wine can detect it and use it if available, or fallback to userspace emulation

Pain Point #2:
Thread synchronization

Example Pattern*



- Multiple producers to multiple consumers
- Each Producer can be consumed by subset of Waiters
- Handle uncontended case well
- Quickly wake up threads
- Exactly eventfd+poll semantics

* Not necessarily a real use case. Didatic example

Differences in the API

- Games are very thread-heavy
- Windows has a very conveninent API: `WaitForMultipleObjects()`
  - Allow a program to wait on ANY of several kind of “objects” (threads, mutexes, events, signals, …)
- In Linux
  - epoll on eventfd
  - Mutexes/signals/conditional variables and userspace emulation

Regular Linux Solutions - eventd

- ESYNC: Wine Implementation of WaitForMultipleObjects using eventfd:
  - Semantics of multiple eventfd fit the problem quite well
- Poll on several file descriptors
  - Cost of sleeping/wakeup of epoll become a bottlenecks
  - High memory usage
  - No fastpath that avoids going into the kernel
- We looked into ways to optimize eventfd but with limited success

Other Linux Solutions

- Remodel for a single trigger (mutex, condition variables)
  - execute the dispatch from a worker thread
  - have worker threads that can pick up any job
  - …
- Doesn’t fit `WaitForMultipleObjects` semantics
- Require more work on userspace (more syscalls, atomic operations)
  - We tried…

Futex Wait Multiple

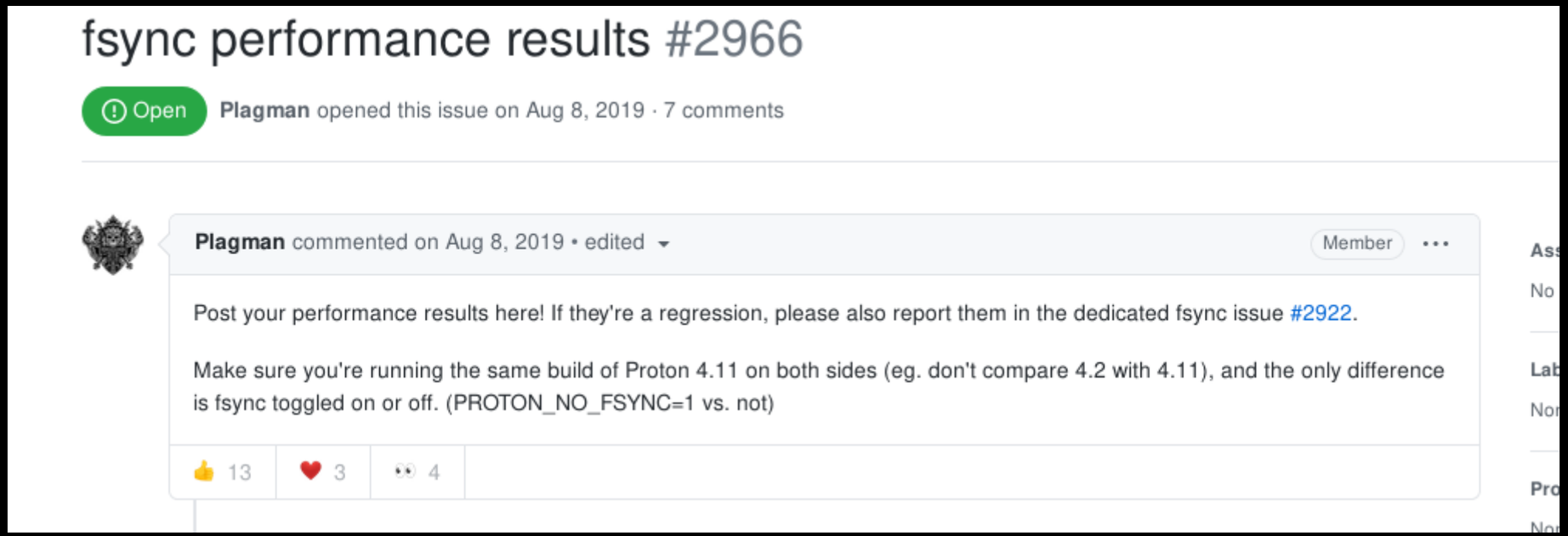
- New Futex operation suggested by Zebediah Figura
- Model the problem well
- Doesn’t completely implement `WaitForMultipleObjects`
  - But allows Wine to implement it with good performance
  - Reuses the idea of a fast path outside of the kernel
- Implemented in Wine by `FSYNC` for supported kernels

Developers feedback

- Positive feedback from the kernel community
- Use-case well understood and acknowledged
- Problems with current futex code
  - Maintainability
  - Complexity
  - Interest in merging incompatible features
- We took the effort to write a new implementation

Users feedback

- Published test kernels and distributed the patch in Liquorix



- Users reported from 1.05x to 2x increase in fps (different games)
- At least one game was really bad on my system without fsync

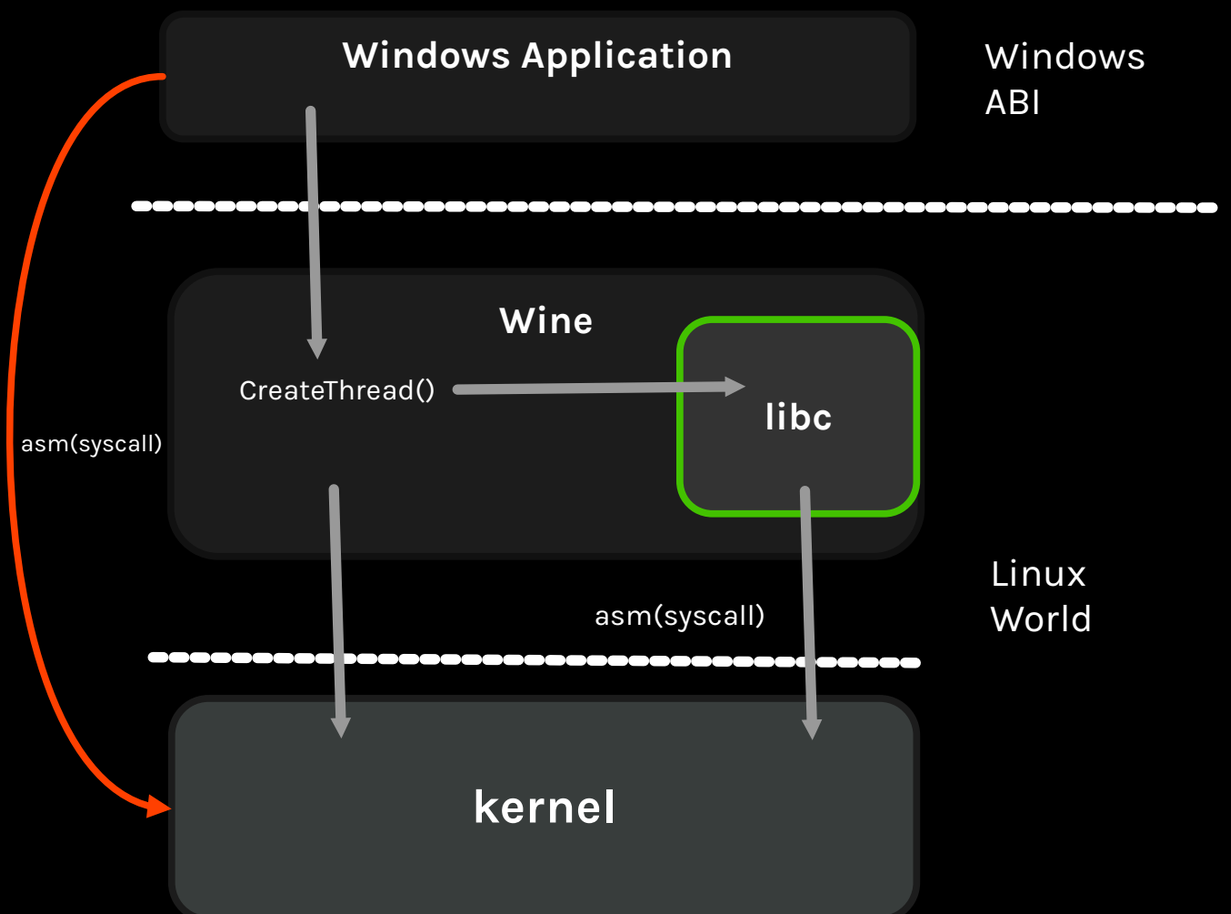
futex2

- Solve longstanding pain points for the kernel community
- Bring other features that will be useful for games/Wine:
  - NUMA awareness
  - Variable sized futexes
- Make vectored futexes a first-class citizen
  - New implementation allow us to tinker more
  - Very promissing results

Pain Point #3:
System call emulation

Modern games DRM & anti-cheat technologies

- Games issuing Windows syscalls directly
- Wine cannot capture them
  - as it go straight to the kernel
- Reaches Linux kernel...
- ... and game crashes
- DRM & anti-cheat

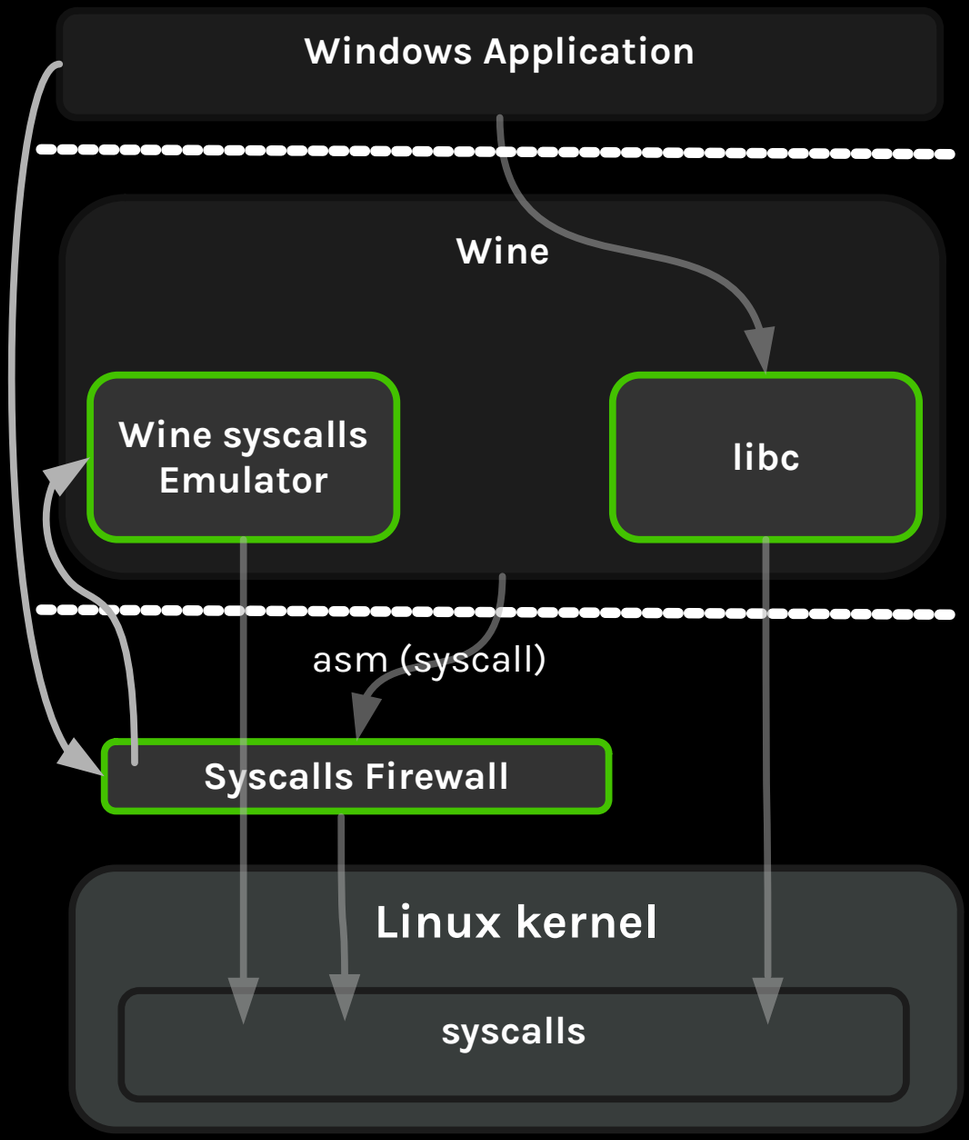


Userspace attempted solutions

- Once again, we usually look for a userspace solution first
  - Dynamically rewrite system call instructions
  - SECCOMP based
- Problems with:
  - Trips anti-cheating software
  - Performance
- What is the right solution?

New mechanism in kernelspace

- Syscall User Dispatch
- Integrates with Seccomp
- Permits quick emulation for compatibility modes
- Useful for other traces
- Expected to land in kernel 5.11
- Distros adoption: Just a CONFIG option



Take-aways

- Other pain-points
- Only extend the kernel when strictly necessary
- Favor non-invasive designs that fit Linux
- Very specific features might be necessary
  - But only when necessary
- We are seeing real performance improvements in games on Linux
- Making Linux support more and more games
  - Natively and emulated

Acknowledgments

- **Valve** and **Pierre-Loup Griffais** for sponsoring and supporting this work.

- Wine developers, in particular **Zebediah Figura** and **Paul Gofman,** for design ideas and a lot of technical input.

Thank you

```
Message {
  config {
    priority: "high"
    body: "Collabora is hiring"  // Many open positions
    recipient: "you"             // Please join us
    calltoaction: "http://col.la/join"
  }
}
```

Gabriel Krisman Bertazi

krisman@collabora.com