Gate project

Timo Savola
FOSDEM 2020

Portable execution state

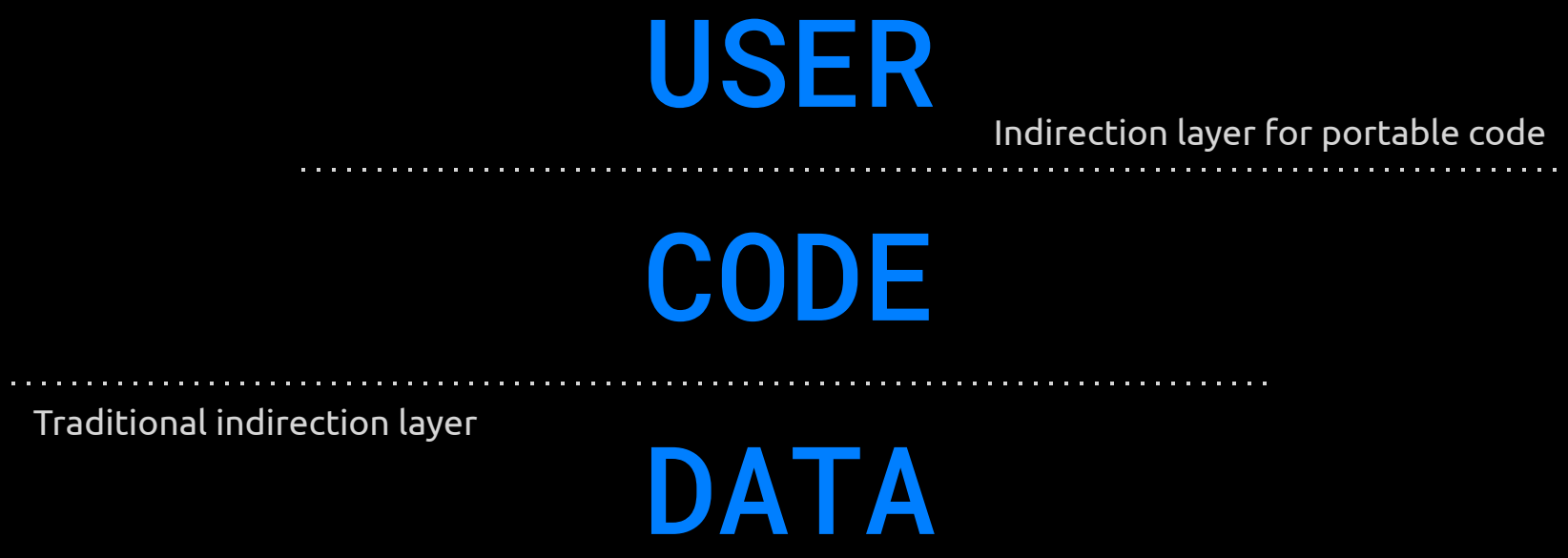
Migrate live programs between desktops, servers and devices - safely.

Gain control by repositioning the abstraction layer.

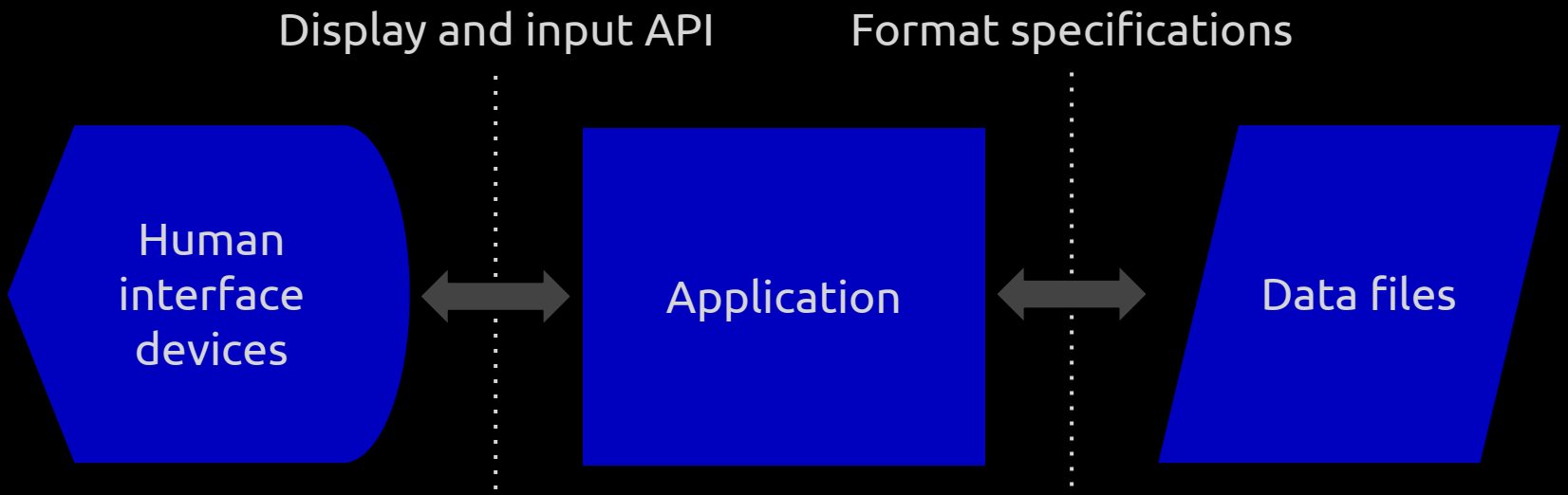
Distributed software architecture, or dynamic network architecture.

Disclaimer: not a blockchain.

Reposition the abstraction layer

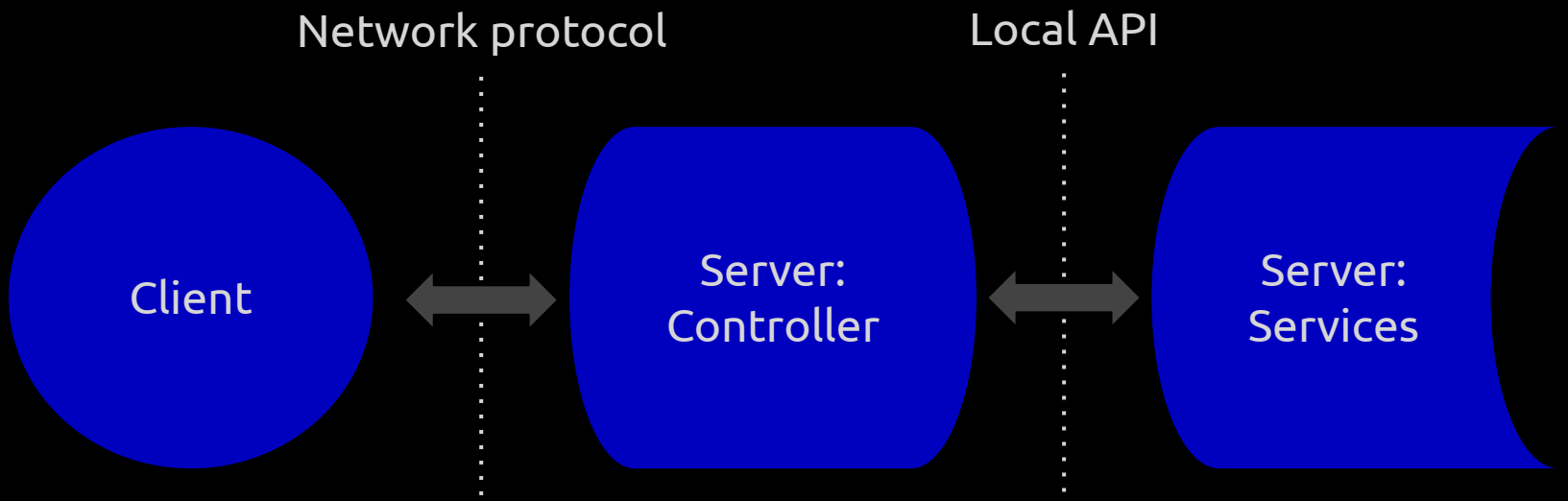


Reduce external interface surface



Data is portable. Portable code can be bundled with it, dissolving the boundary.

Reposition communication interface for locality



API can be moved *into* the server. Network I/O patterns become a client detail.

Gate

Personal hobby research project.

In development for 5 years - or over 10 if counting previous experiments.

BSD license.

https://github.com/tsavola/gate
https://gate.computer

Three tiers

- WebAssembly
  - Portable program format, and a tooling ecosystem to go along with it.
- Runtime for untrusted code
  - Usual Linux containerization features, but with extreme decoupling.
- Pluggable, discoverable services
  - Hosts can provide their own sets of APIs.

Portable snapshot and restore

No support needed from user programs.

A running instance can be suspended at any time. The effect is immediate (or at least the time is bounded).

Snapshots are WebAssembly binaries with Gate-specific custom sections. Other runtimes could load them, but they appear as modules without any export functions.

Halted instances have returned from their entry function. Such snapshots have export functions, which may be called to re-enter the program.

Internals

Go packages, including a WebAssembly compiler:

```
https://github.com/tsavola/wag
```

Runtime core implemented in C and assembly.

Implementation is currently Linux-specific. Supports x86-64 and ARM64.

Can also run on Android.

Safety

WebAssembly defines a logical sandbox.

Each program invocation has its own OS process.

Service interaction happens via IPC messages sent through pipes.

Linux syscalls restricted via seccomp filter:

> Whitelist: read, write, close, ppoll, mprotect, rt_sigreturn, exit_group.
> mprotect arguments are restricted.

Finally, employ all the Linux namespaces to protect the host system.

Services

Services are discovered and may disappear as the program migrates.

Implementations:

- `catalog` – explore available services.
- `origin` – I/O with the originator/owner of the instance (≈ stdio).
- `gate.computer/localhost` – access whitelisted HTTP endpoints.
- …

Services are implemented in Go. State serialization has an important role.

Next step: Support communication among peers on a server.

User program APIs

Impossible to support standard APIs meaningfully. Limited WASI support; Gate services are accessible through a dedicated file descriptor.

No blocking system calls. Purely asynchronous programming model.

Primitive C API. Used for simple test programs.

Rust is ideal for lightweight WebAssembly programs:

- Gain crate provided Gate support, but it's out of date.
- Next step: Update it, with std futures and async/await syntax support.

Demo

1. Start the Gate port of Doom on an x86-64 machine.
2. Suspend it (SIGQUIT).
3. Show stack trace at the suspension point.
4. Create a snapshot.
5. Inspect the snapshot using wasm-objdump.
6. Copy the snapshot to an ARM64 machine.
7. Resume the game from the snapshot.

https://github.com/tsavola/doom
https://gate.computer/raster

```
x86-64 $ uname -a
Linux saukko 5.3.0-28-generic #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
x86-64 $ gate call doom.wasm < /usr/share/games/doom/doom1.wad
                            DOOM Shareware Startup v1.10
V_Init: allocate screens.
M_LoadDefaults: Load system defaults.
Z_Init: Init zone memory allocation daemon.
W_Init: Init WADfiles.
 adding DOOMWADDIR/doom1.wad
===============================================================================
                                Shareware!
===============================================================================
M_Init: Init miscellaneous info.
R_Init: Init DOOM refresh daemon - [..                ]
InitTextures
InitFlats........
InitSprites
InitColormaps
R_InitData
R_InitPointToAngle
R_InitTables
R_InitPlanes
R_InitLightTables
R_InitSkyMap
R_InitTranslationsTables
P_Init: Init Playloop state.
I_Init: Setting up machine state.
D_CheckNetGame: Checking network game status.
startskill 2  deathmatch: 0  startmap: 1  startepisode: 1
player 1 of 1 (1 nodes)
S_Init: Setting up sound.
HU_Init: Setting up heads up display.
ST_Init: Init status bar.
```

```
x86-64 $ gate call doom.wasm < /usr/share/games/doom/doom1.wad
                           DOOM Shareware Startup v1.10
V_Init: allocate screens.
M_LoadDefaults: Load system defaults.
Z_Init: Init zone memory allocation daemon.
W_Init: Init WADfiles.
 adding DOOMWADDIR/doom1.wad
===============================================================================
                               Shareware!
===============================================================================
M_Init: Init miscellaneous info.
R_Init: Init DOOM refresh daemon - [..                ]
InitTextures
InitFlats........
InitSprites
InitColormaps
R_InitData
R_InitPointToAngle
R_InitTables
R_InitPlanes
R_InitLightTables
R_InitSkyMap
R_InitTranslationsTables
P_Init: Init Playloop state.
I_Init: Setting up machine state.
D_CheckNetGame: Checking network game status.
startskill 2  deathmatch: 0  startmap: 1  startepisode: 1
player 1 of 1 (1 nodes)
S_Init: Setting up sound.
HU_Init: Setting up heads up display.
ST_Init: Init status bar.
^\
bc32807d-eee8-4775-b4dd-48abdee67bfc SUSPENDED
x86-64 $
```

```
x86-64 $ gate snapshot bc32807d-eee8-4775-b4dd-48abdee67bfc snapshot.wasm
M4nu1fwg81A-SHsc27CGi91yjliyu67VDTev12N-s5VkCIpUCzwk8aUoF6IDbuJ1
x86-64 $ gate debug bc32807d-eee8-4775-b4dd-48abdee67bfc backtrace
#0  0x6e69 in NetUpdate at /home/user/doom/linuxdoom-1.10/d_net.c:320
#1  0x7e1e in TryRunTics at /home/user/doom/linuxdoom-1.10/d_net.c:655
       0  0000000000000004 0000000000000001 0000000000000c0c 0000000000000001
       4  0000000000000be8 0000000000000001 0000000000000be8 0000000000000000
#2  0x5434 in D_DoomLoop at /home/user/doom/linuxdoom-1.10/d_main.c:386
       0  0000000000000000
#3  0x6aaf in D_DoomMain at /home/user/doom/linuxdoom-1.10/d_main.c:0
       0  0000000001922450 0000000000000000 0000000000000000 0000000000000000
       4  0000000000000000
#4  0x0896 in _start at /home/user/doom/linuxdoom-1.10/libc.c:262
x86-64 $
```

```
x86-64 $ wasm-objdump -h snapshot.wasm

snapshot.wasm:  file format wasm 0x1

Sections:

     Type start=0x0000000b end=0x000000a3 (size=0x00000098) count: 20
   Import start=0x000000a5 end=0x00000105 (size=0x00000060) count: 4
 Function start=0x00000108 end=0x00000364 (size=0x0000025c) count: 602
    Table start=0x00000366 end=0x0000036d (size=0x00000007) count: 1
   Memory start=0x0000036f end=0x00000373 (size=0x00000004) count: 1
   Global start=0x00000375 end=0x0000037e (size=0x00000009) count: 1
   Custom start=0x00000380 end=0x00000399 (size=0x00000019) "gate.snapshot"
   Custom start=0x0000039b end=0x000003bc (size=0x00000021) "gate.export"
     Elem start=0x000003bf end=0x000004d7 (size=0x00000118) count: 1
     Code start=0x000004db end=0x00033faf (size=0x00033ad4) count: 602
   Custom start=0x00033fb1 end=0x00033ff4 (size=0x00000043) "gate.buffer"
   Custom start=0x00033ff7 end=0x000340a2 (size=0x000000ab) "gate.stack"
     Data start=0x000340a7 end=0x019640b0 (size=0x01930009) count: 1
   Custom start=0x019640b4 end=0x019e1e2c (size=0x0007dd78) ".debug_info"
   Custom start=0x019e1e30 end=0x019f2382 (size=0x00010552) ".debug_loc"
   Custom start=0x019f2386 end=0x019f64a4 (size=0x0000411e) ".debug_ranges"
   Custom start=0x019f64a8 end=0x019fd7f1 (size=0x00007349) ".debug_abbrev"
   Custom start=0x019fd7f5 end=0x01a29f02 (size=0x0002c70d) ".debug_line"
   Custom start=0x01a29f06 end=0x01aba44d (size=0x00090547) ".debug_str"
   Custom start=0x01aba450 end=0x01abc9b5 (size=0x00002565) "name"
   Custom start=0x01abc9b7 end=0x01abca22 (size=0x0000006b) "producers"
x86-64 $
```

```
arm64 $ uname -a
Linux graviton 4.15.0-1057-aws #59-Ubuntu SMP Wed Dec 4 09:58:16 UTC 2019 aarch64 aarch64 aarch64 GNU/Linux
arm64 $ gate call snapshot.wasm
```

Gate components

`gate`

Command-line client for the local daemon and remote servers.

`gated`

D-Bus daemon running and managing programs for the local user.

`gate-server`

Web server serving the public, or just authenticated users.

Server highlights

Can be configured to serve anonymous drive-by execution requests.

Uses Ed25519 public keys for grouping persistent resources.
Authentication is optional. Supports SSH keys and authorized_keys files.

Optional IPFS support for sourcing programs.

Remote WebAssembly debugging with breakpoints. Portable snapshots.

Program and instance image management

Stored in sparse files; snapshotting requires shared memory mappings.

Backends:

- memfd (or ashmem on Android).
- Regular files on a filesystem, optimized for zero-copy (reflink).

Normally, programs and suspended instances would go on the filesystem, and running instances in memory. But instances can also be directly backed by the filesystem.

WebAssembly “microcode”

Additional safety layer. Written in WebAssembly text format for stability.

Trusted WebAssembly library between user code and low-level runtime functions (syscall wrappers) implemented in x86-64/ARM64 assembly.

Implements the Gate runtime ABI (including WASI). Pointer arguments of ABI functions need to be checked carefully before accessing memory.

The low-level functions avoid pointers so that the WebAssembly compiler can generate checked memory access code outside of hand-written assembly code.

gate.computer

savo.la