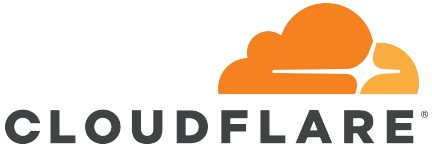


SRECON ASIA AUSTRALIA

Call to ARMs:

adopting an arm64 server into x86 infrastructure

Ignat Korchagin

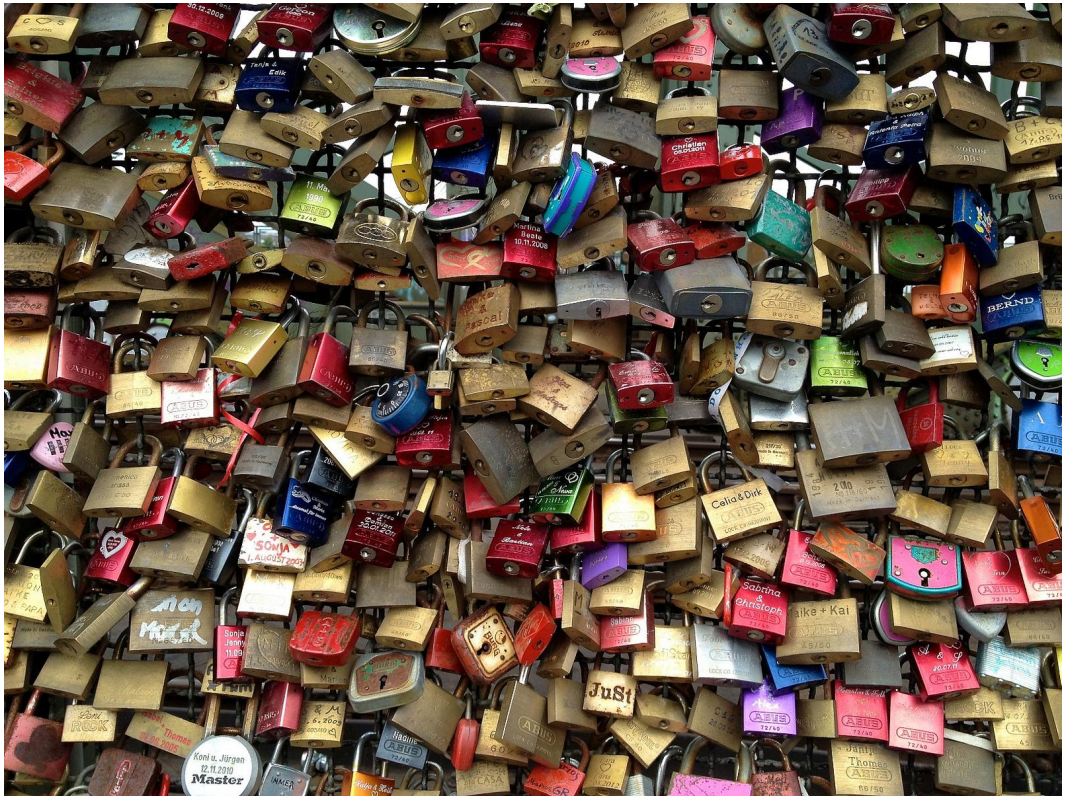
@secumod

$ whoami

- Platform engineer at Cloudflare
- Passionate about security and crypto
- Enjoy low level programming

Why?

Vendor lock-in

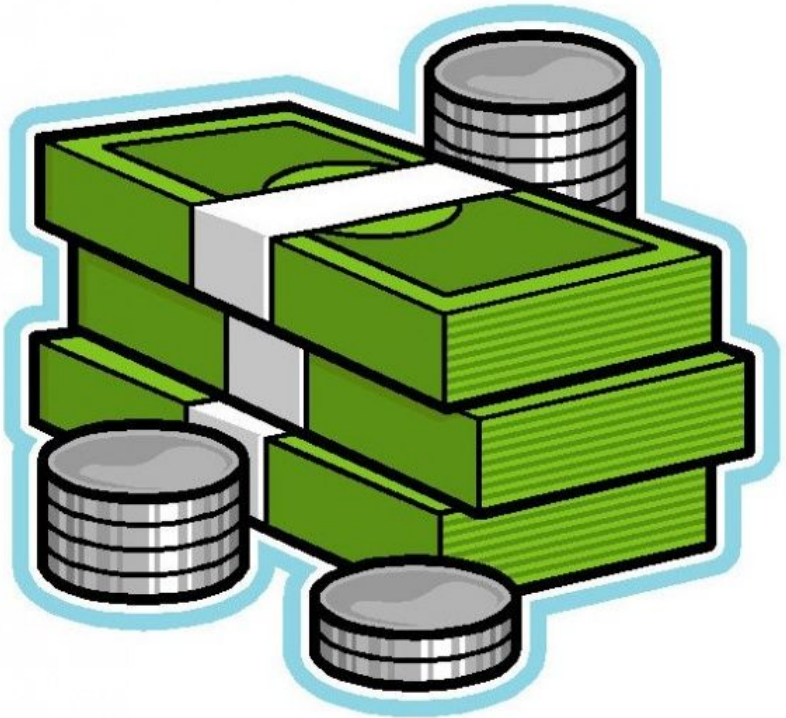


CLOUDFLARE

Save the power

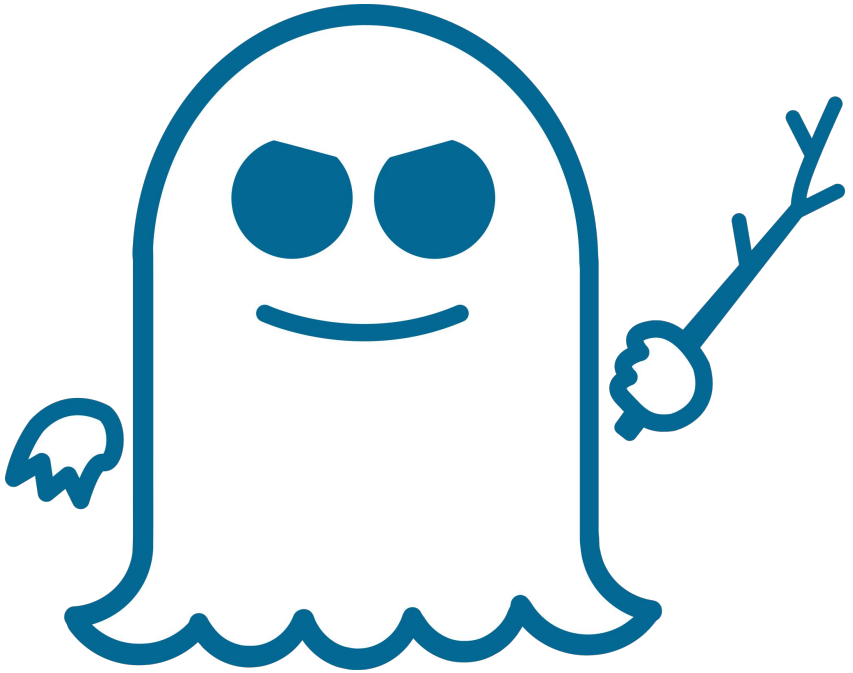
CLOUDFLARE

Cut equipment costs



CLOUDFLARE

Security



CLOUDFLARE

Why ARM64?

CLOUDFLARE

Why ARM64?

- performs well in the mobile/IoT space

Why ARM64?

- performs well in the mobile/IoT space
- potentially more power-efficient

Why ARM64?

- performs well in the mobile/IoT space
- potentially more power-efficient
- huge developer community

Why ARM64?

- performs well in the mobile/IoT space
- potentially more power-efficient
- huge developer community
- first class support in Linux

Why ARM64?

- performs well in the mobile/IoT space
- potentially more power-efficient
- huge developer community
- first class support in Linux
- established tools

Why ARM64?

- performs well in the mobile/IoT space
- potentially more power-efficient
- huge developer community
- first class support in Linux
- established tools
- > 32 bits

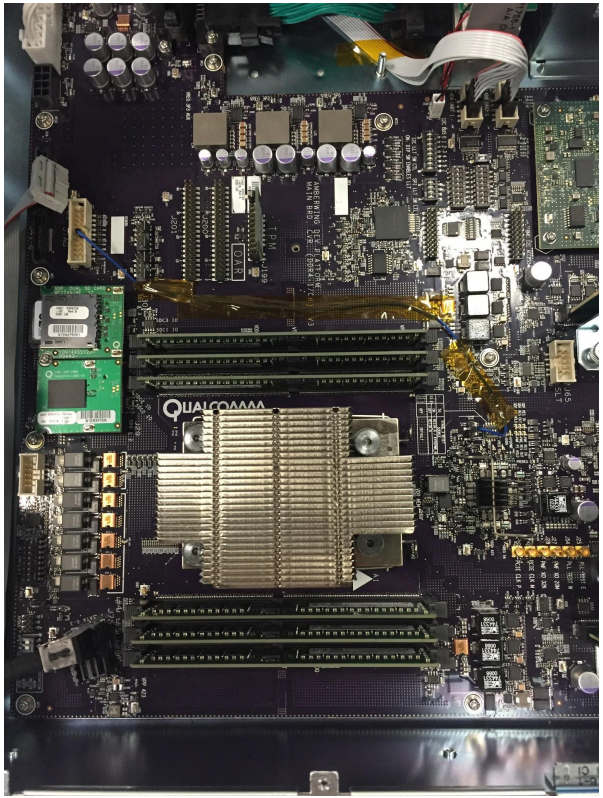
Why ARM64?

- performs well in the mobile/IoT space
- potentially more power-efficient
- huge developer community
- first class support in Linux
- established tools
- > 32 bits
- mitigates the RISC ;)

Initial benchmarks

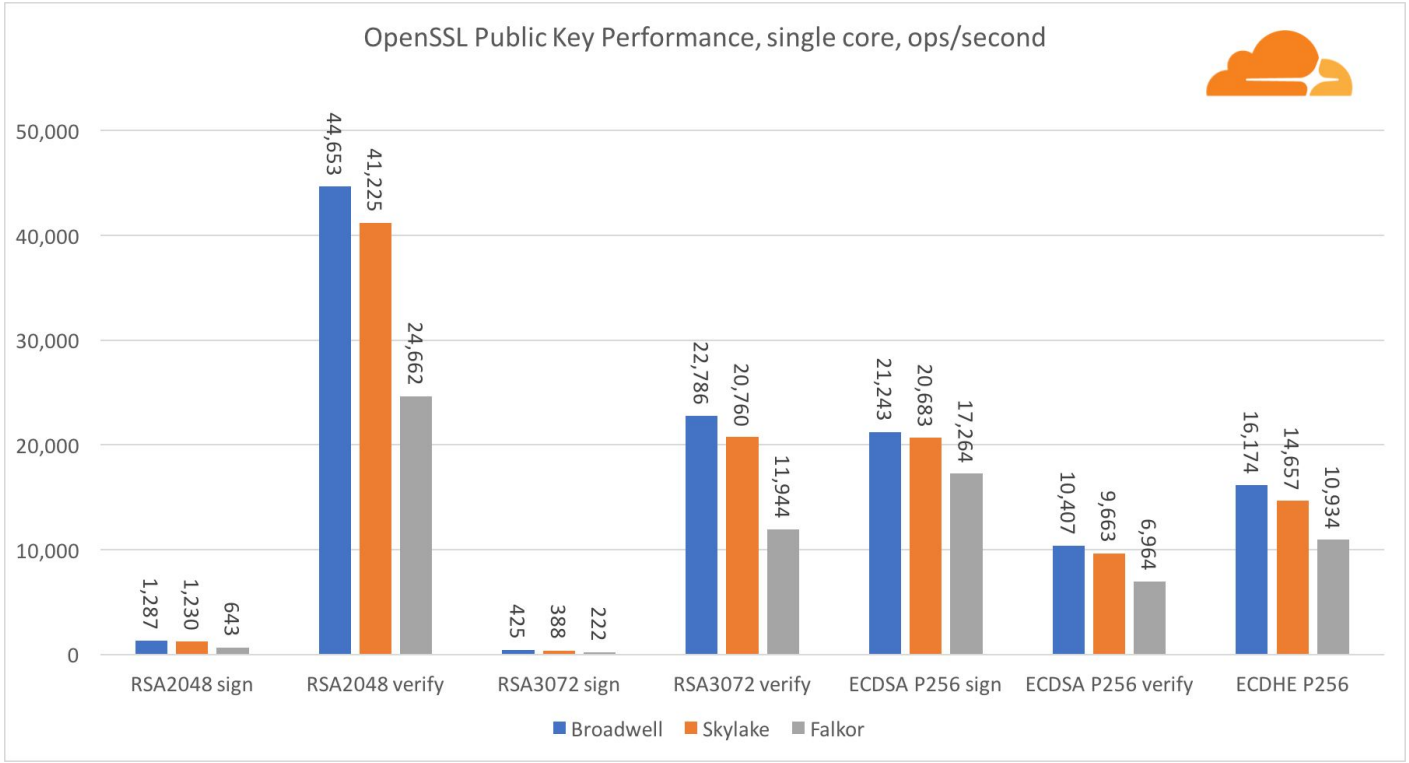
So you have an ARM64 server

- Falkor core
- 46 cores
- 2.5 GHz
- Thermal design power 120W
  - compared to 170W Skylake



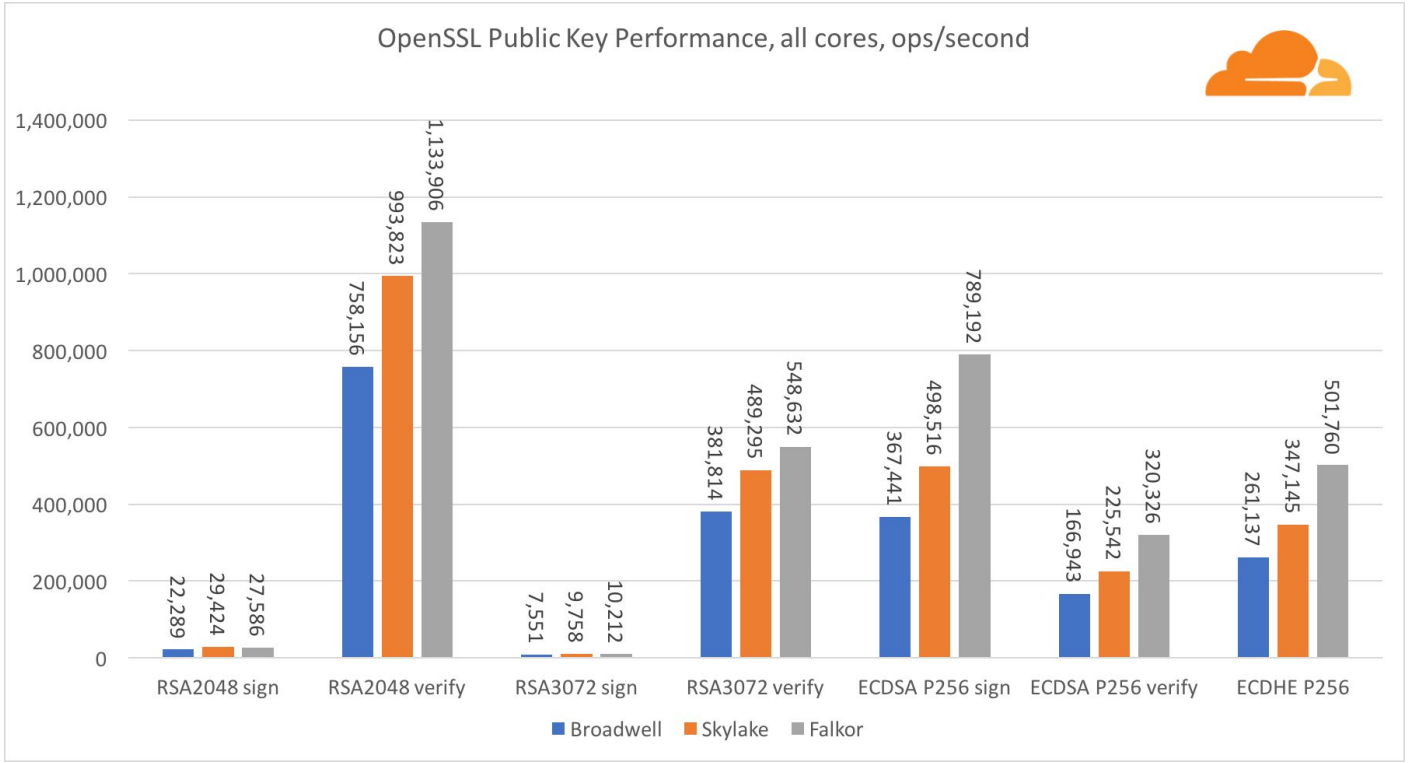
CLOUDFLARE

Public key cryptography (single core)



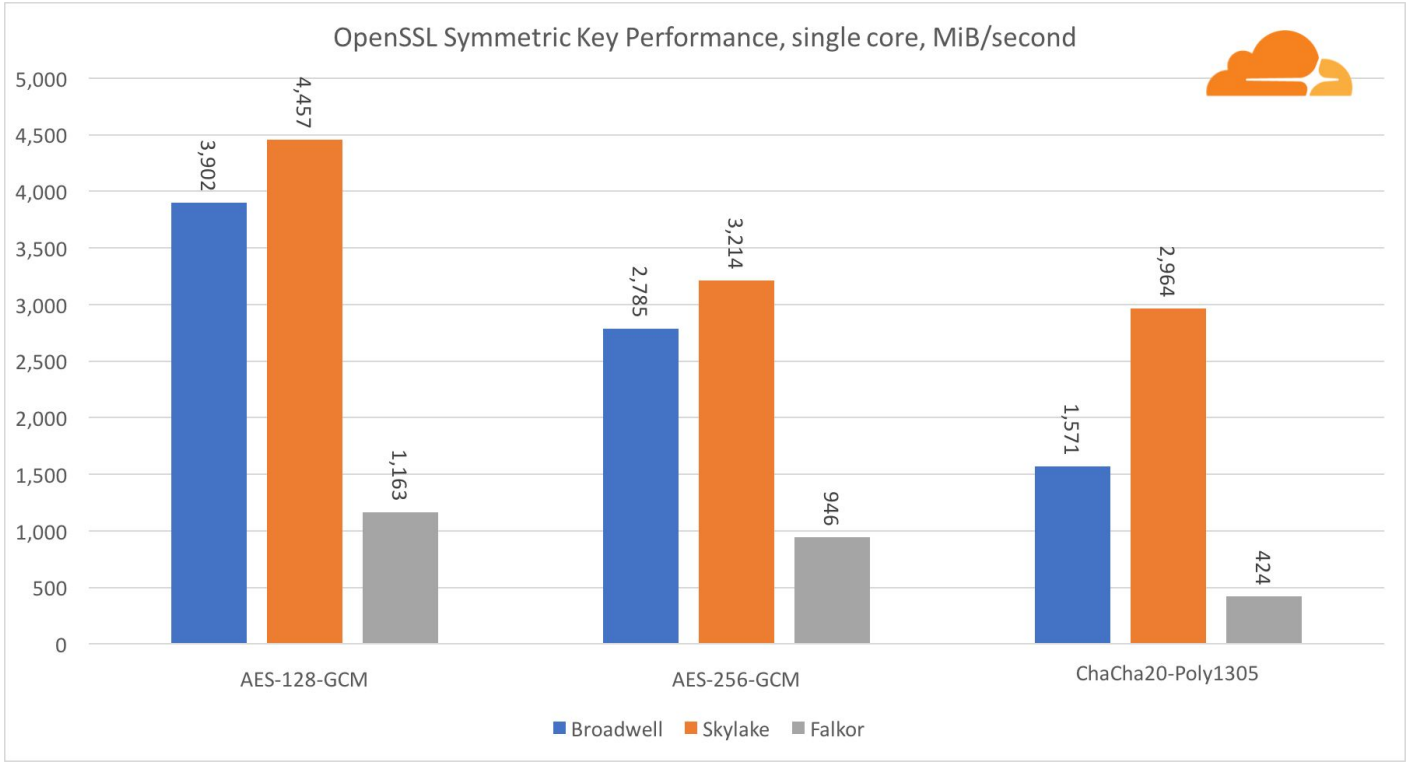
CLOUDFLARE

Public key cryptography (all cores)



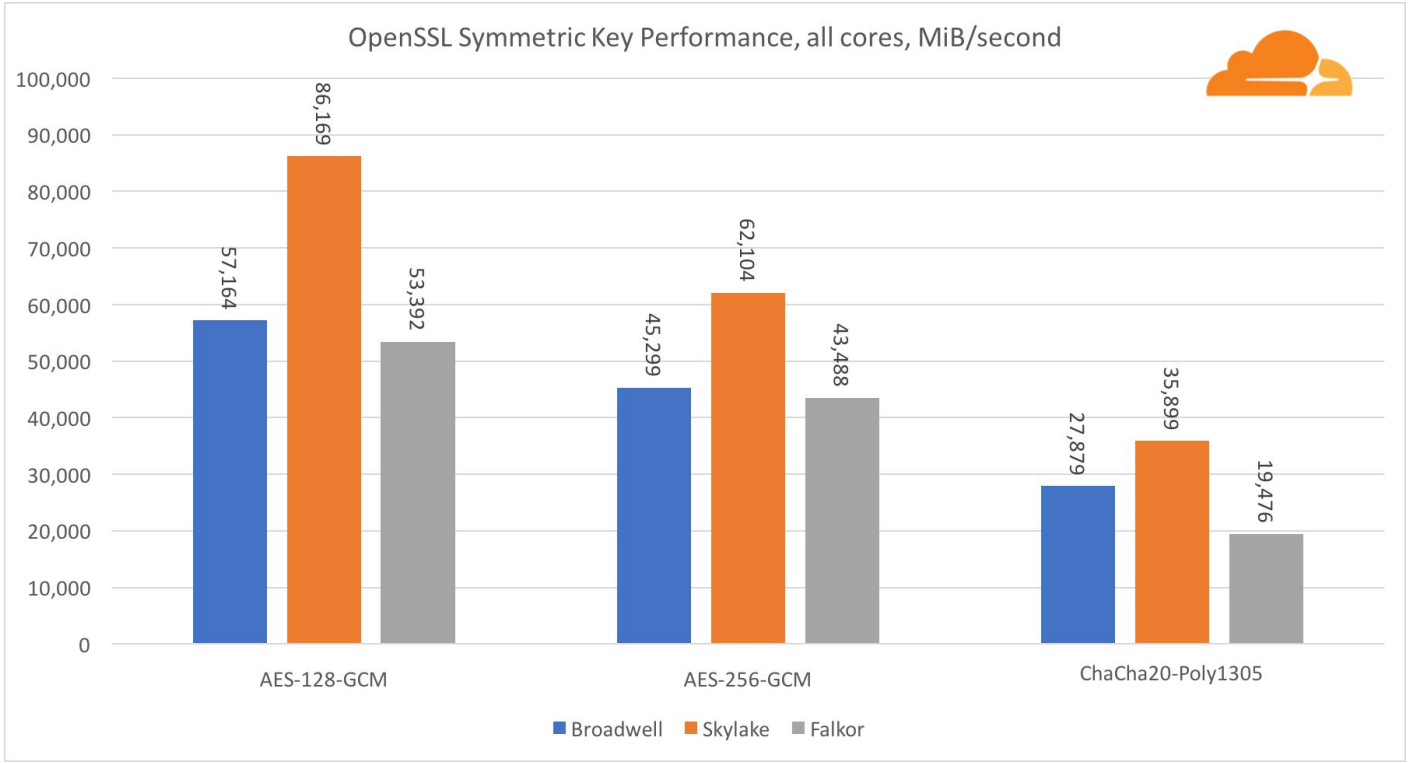
CLOUDFLARE

Symmetric cryptography (single core)



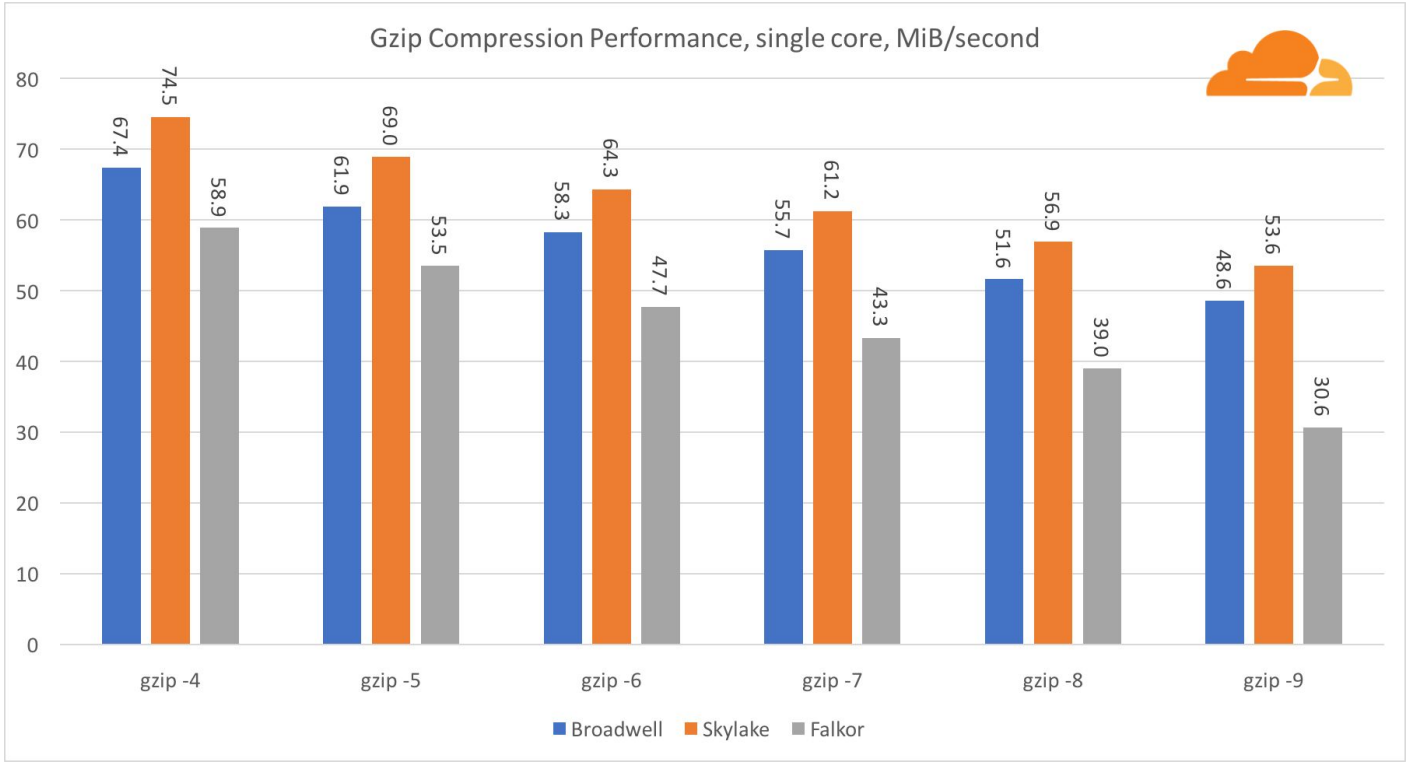
CLOUDFLARE

Symmetric cryptography (all cores)



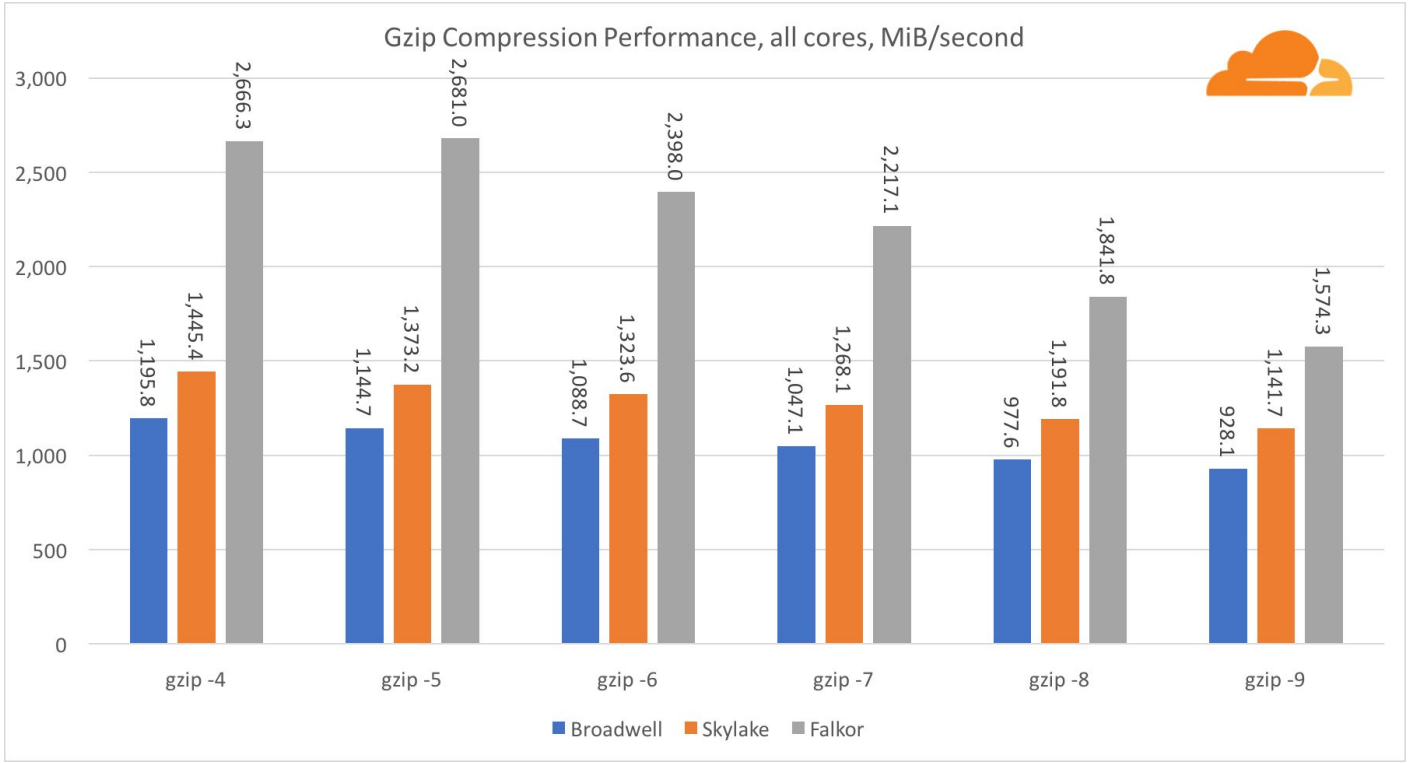
CLOUDFLARE

Gzip (single core)



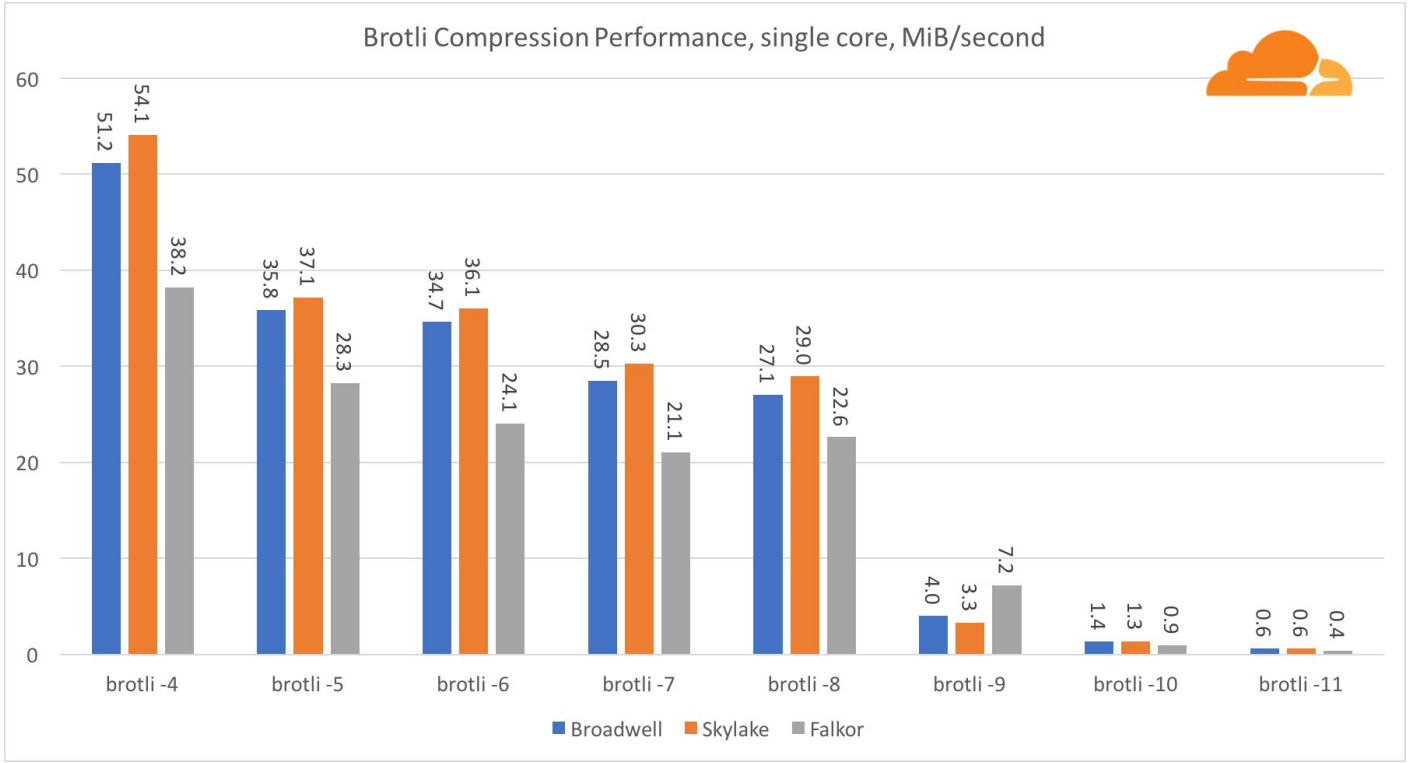
CLOUDFLARE

Gzip (all cores)



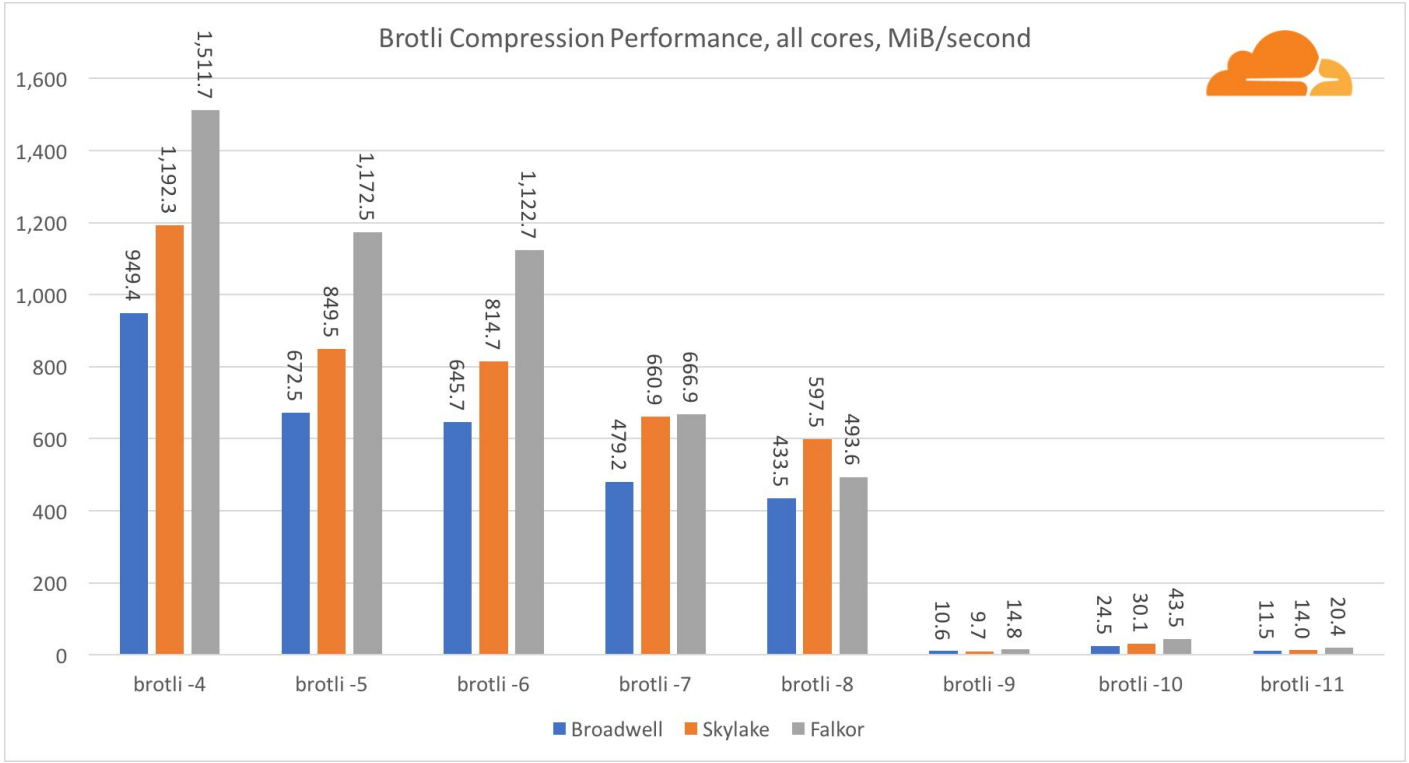
CLOUDFLARE

Brotli (single core)



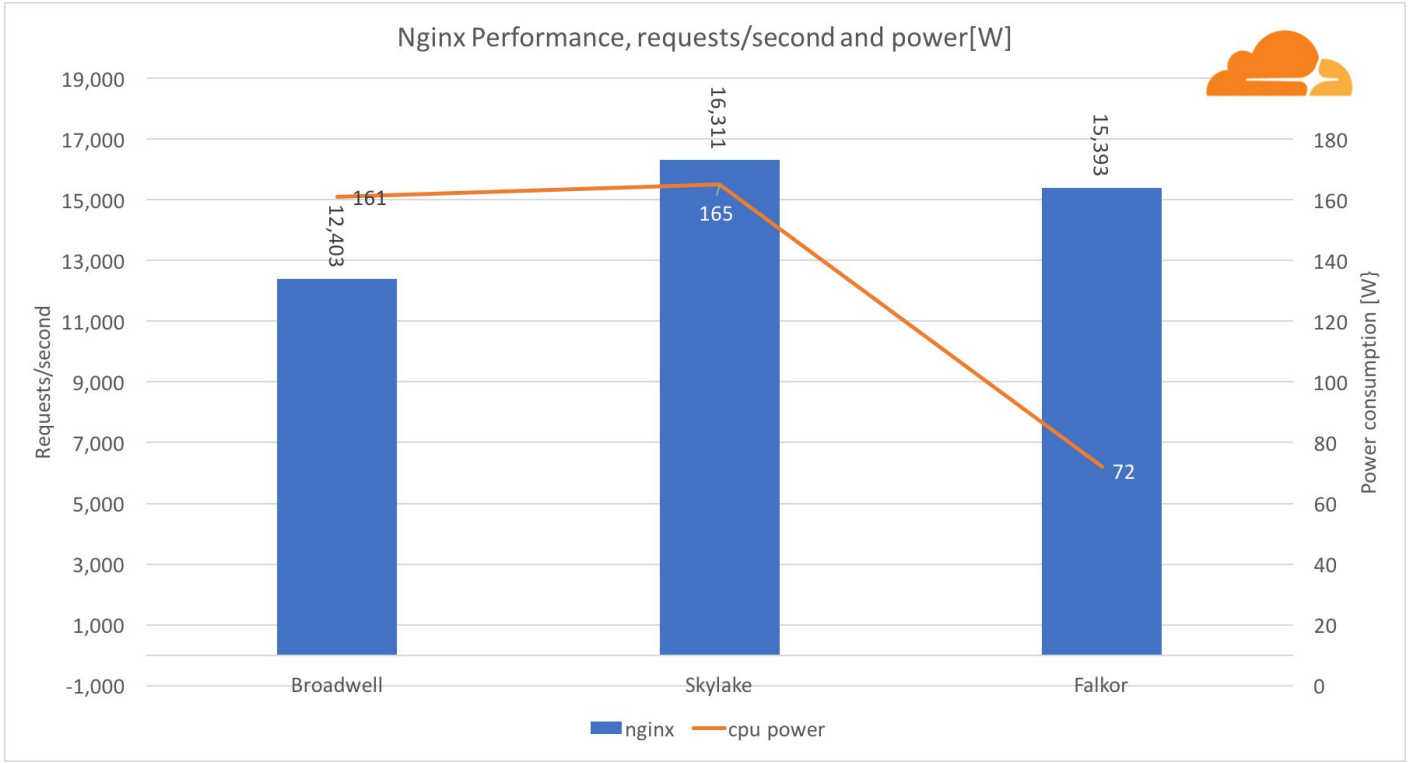
CLOUDFLARE

Brotli (all cores)



CLOUDFLARE

nginx (with power!)



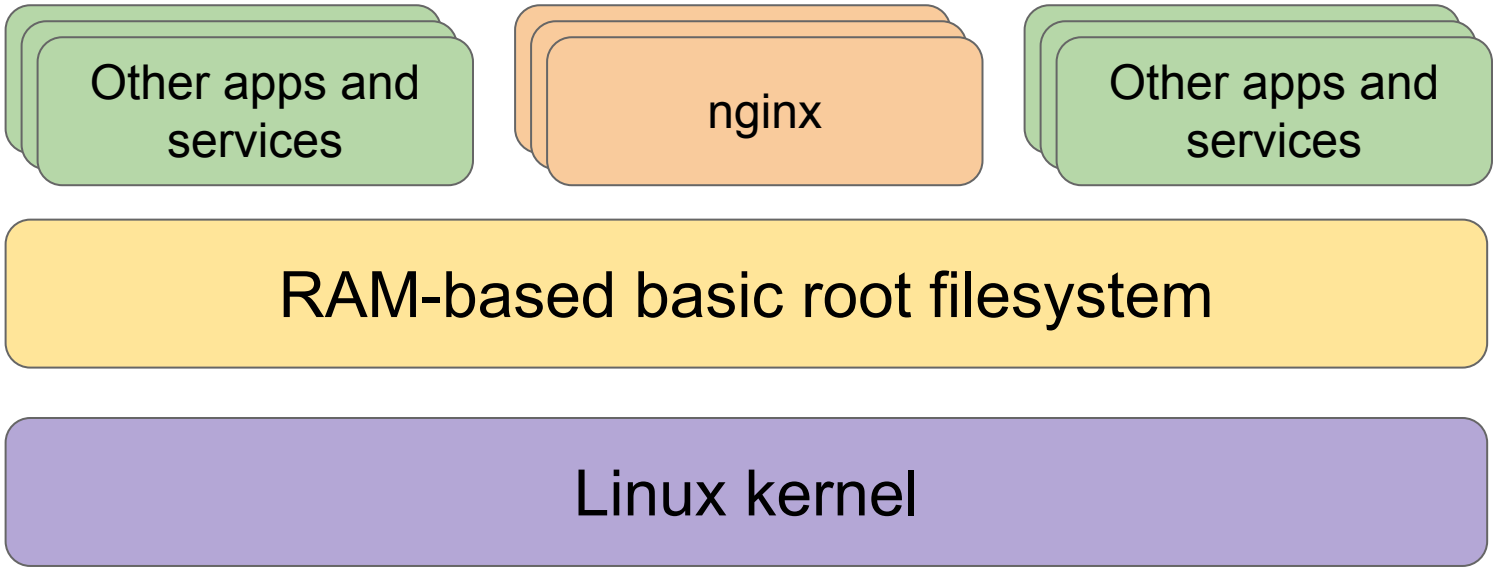
CLOUDFLARE

Putting an ARM64 server in a DC

Initial integration in the DC

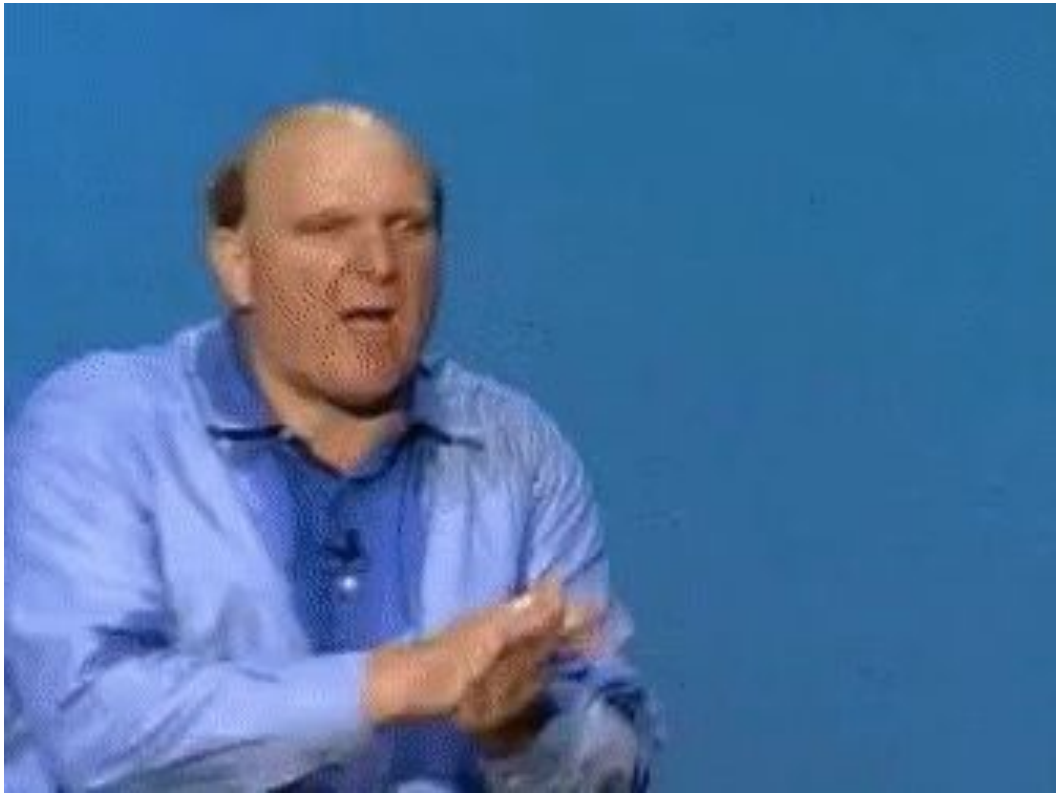
CLOUDFLARE

Edge server software stack



CLOUDFLARE

Consider your developers



CLOUDFLARE

Building packages for ARM64

production arch != developer arch

Building packages for ARM64

production arch != developer arch

- need to (cross-)compile packages for a different architecture

Building packages for ARM64

production arch != developer arch

- need to (cross-)compile packages for a different architecture
- cannot run even basic unit tests locally

Building packages for ARM64 options

- cross-compiling
  - relatively easy, but requires many changes
  - potential side-effects (ex. library paths)

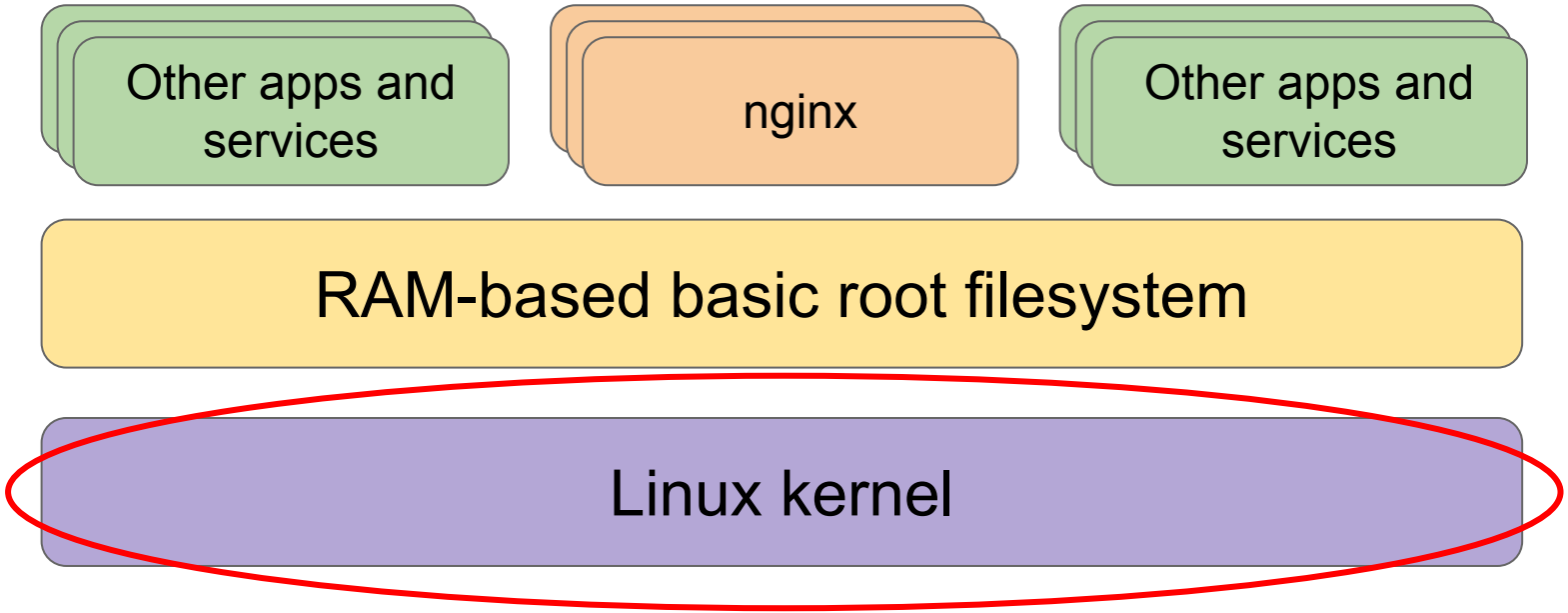
Building packages for ARM64 options

- cross-compiling
  - relatively easy, but requires many changes
  - potential side-effects (ex. library paths)
- native builds on arm64 servers
  - no spare hardware
  - chicken-and-egg problem: requires setting up an arm64 server and we need packages for it

Building packages for ARM64 options

- cross-compiling
  - relatively easy, but requires many changes
  - potential side-effects (ex. library paths)
- native builds on arm64 servers
  - no spare hardware
  - chicken-and-egg problem: requires setting up an arm64 server and we need packages for it
- ???

Edge server software stack



CLOUDFLARE

Cross-compiling Linux kernel

- need a cross-compiler
  - sudo apt-get install crossbuild-essential-arm64

Cross-compiling Linux kernel

- need a cross-compiler
  - sudo apt-get install crossbuild-essential-arm64
- need to adjust kernel build system to call the cross-compiler
  - make (xxxconfig) => make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- O=arm64build (xxxconfig)

Cross-compiling Linux kernel

- need a cross-compiler
  - sudo apt-get install crossbuild-essential-arm64
- need to adjust kernel build system to call the cross-compiler
  - make (xxxconfig) => make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- O=arm64build (xxxconfig)
- need a working kernel configuration file
  - cp config-amd64 .config
  - make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- oldconfig

ARM64 Linux kernel caveats

- need to trace down and install many arm64 build dependencies for perf
  - otherwise perf is very limited

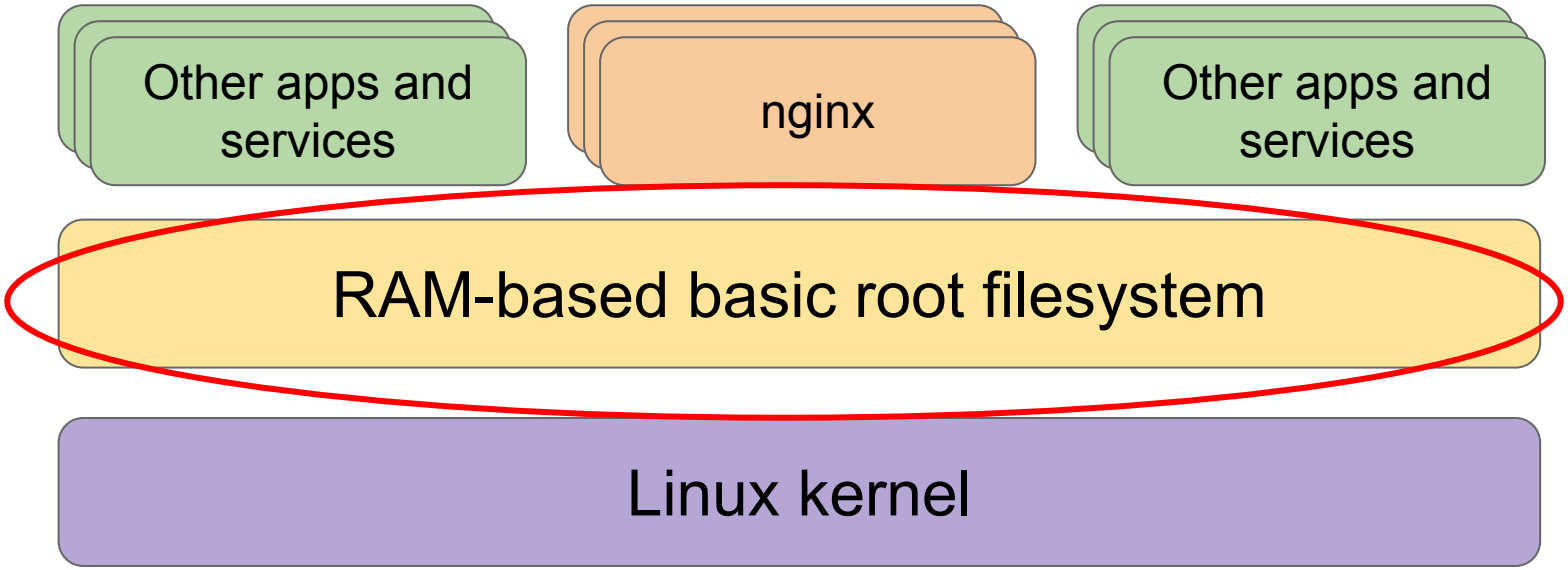
ARM64 Linux kernel caveats

- need to trace down and install many arm64 build dependencies for perf
  - otherwise perf is very limited
- need to trace down and enable required hardware modules
  - our OOB console access did not work until we enabled some non-standard serial driver

ARM64 Linux kernel caveats

- need to trace down and install many arm64 build dependencies for perf
  - otherwise perf is very limited
- need to trace down and enable required hardware modules
  - our OOB console access did not work until we enabled some non-standard serial driver
- by default your kernel will be configured with 39-bit virtual address space
  - allows to address up to 512GB
  - https://www.kernel.org/doc/Documentation/arm64/memory.txt

Edge server software stack



CLOUDFLARE

ARM64 baseimg

- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs

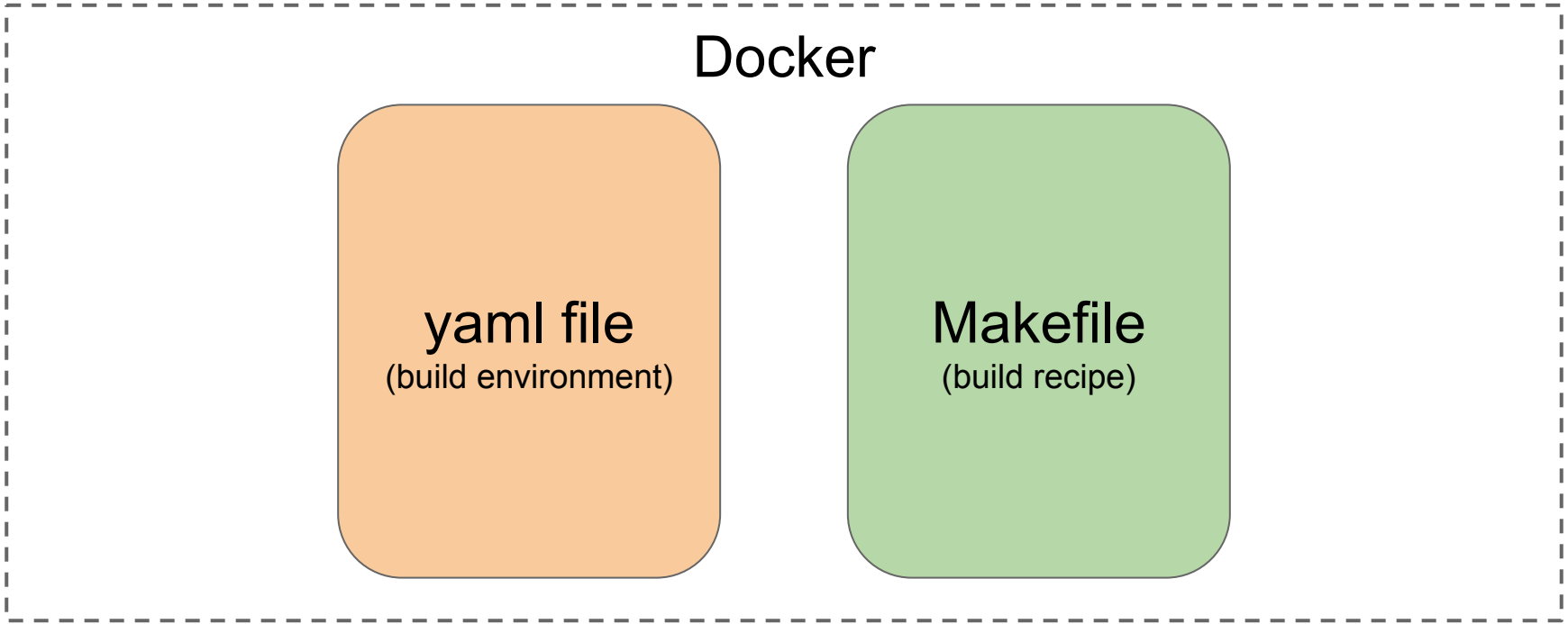
ARM64 baseimg

- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs
- no easy way to do cross-arch debootstrap
  - https://wiki.debian.org/EmDebian/CrossDebootstrap

ARM64 baseimg

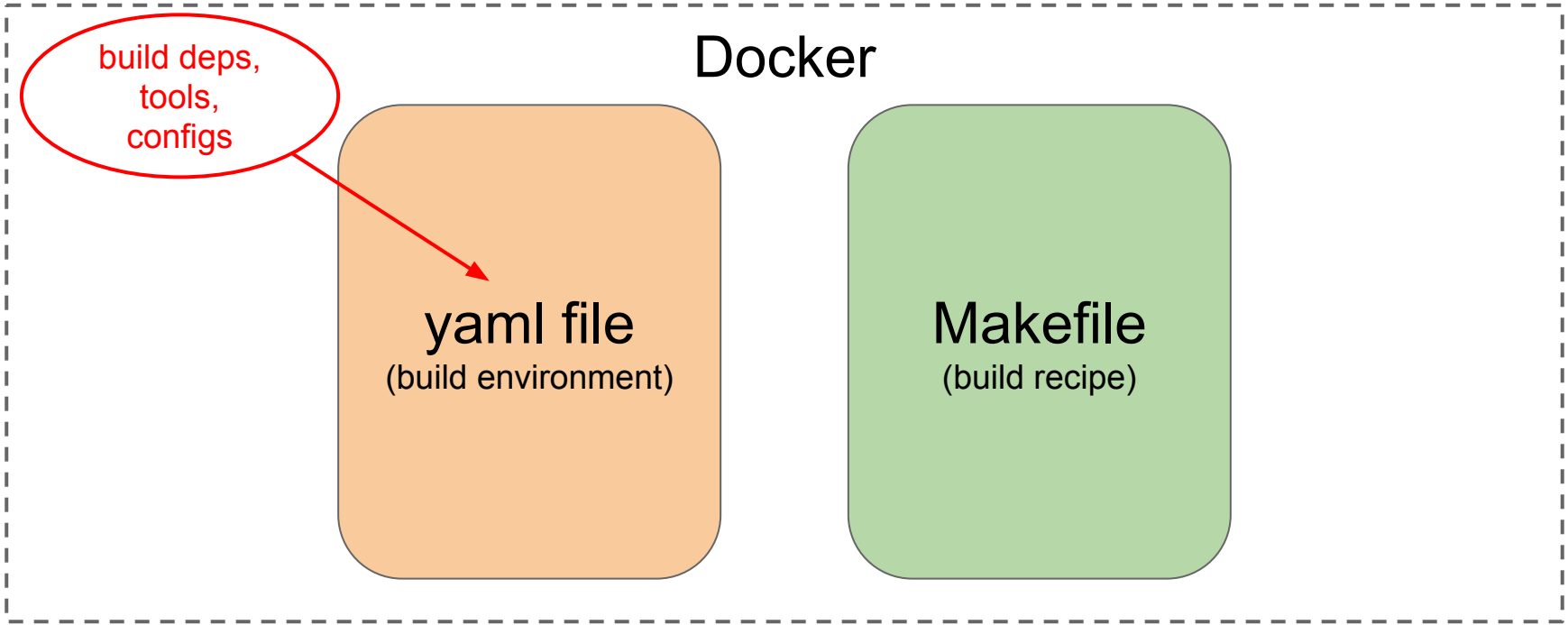
- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs
- no easy way to do cross-arch debootstrap
  - https://wiki.debian.org/EmDebian/CrossDebootstrap
- requires many changes and complex logic in both the build system and build environment

Cloudflare build system



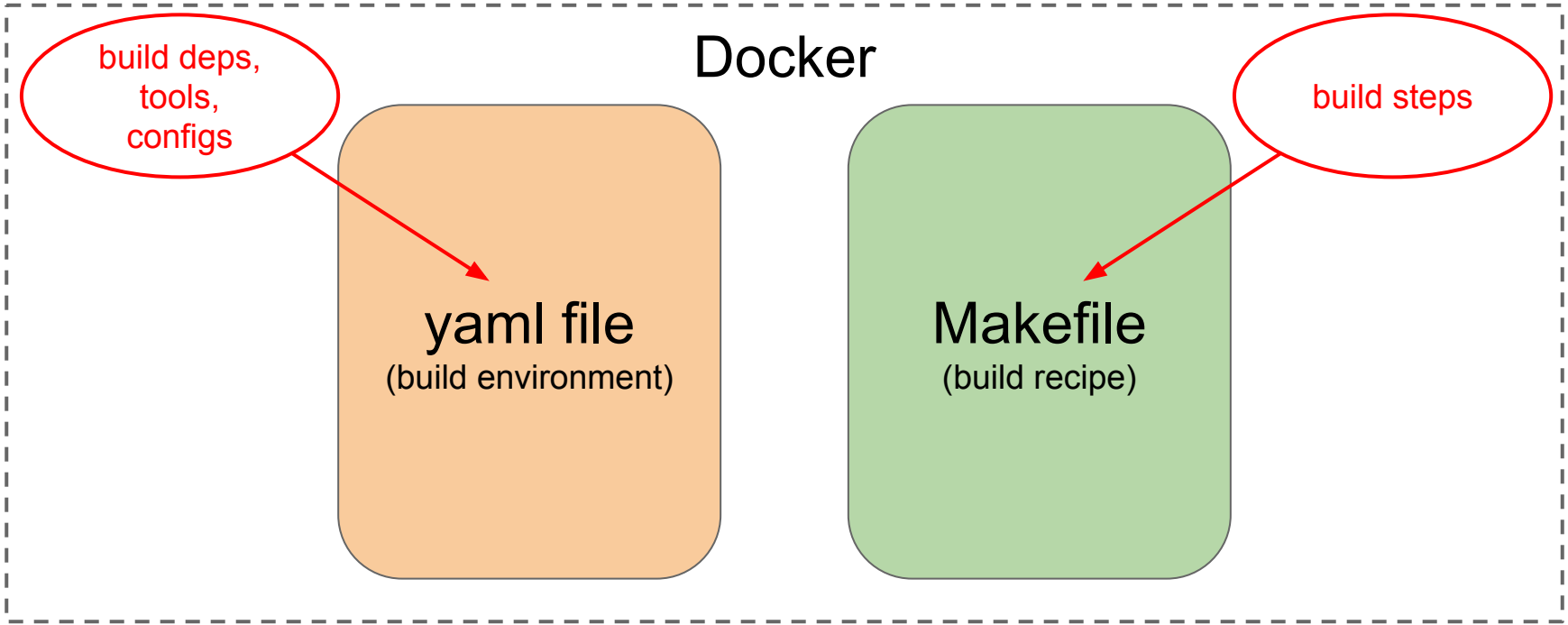
CLOUDFLARE

Cloudflare build system



CLOUDFLARE

Cloudflare build system



CLOUDFLARE

Cloudflare build system

- Docker-based
  - a Makefile - how to build
  - a yaml file - build dependencies and environment

CLOUDFLARE

Cloudflare build system

- Docker-based
  - a Makefile - how to build
  - a yaml file - build dependencies and environment
- reproducible in CI

CLOUDFLARE

Cloudflare build system

- Docker-based
  - a Makefile - how to build
  - a yaml file - build dependencies and environment
- reproducible in CI

- can we emulate ARM64 environment with Docker?
  - without virtual machines
  - change only the build environment, not the recipe

CLOUDFLARE

QEMU user emulation

- dynamically translates foreign architecture code upon execution
- allows to execute arm64 binary directly on x86

qemu-user in action

```
ignat@dev:~$ gcc -static -o helloarch helloarch.c
ignat@dev:~$ readelf -h helloarch | grep -i machine
  Machine:                           Advanced Micro Devices X86-64
ignat@dev:~$ ./helloarch
Hello, x86_64!
ignat@dev:~$ aarch64-linux-gnu-gcc -static -o helloarch helloarch.c
ignat@dev:~$ readelf -h helloarch | grep -i machine
  Machine:                           AArch64
ignat@dev:~$ ./helloarch
bash: ./helloarch: cannot execute binary file: Exec format error
ignat@dev:~$ sudo apt-get install qemu-user-static
ignat@dev:~$ qemu-aarch64-static ./helloarch
Hello, aarch64!
```

binfmt_misc Linux kernel module

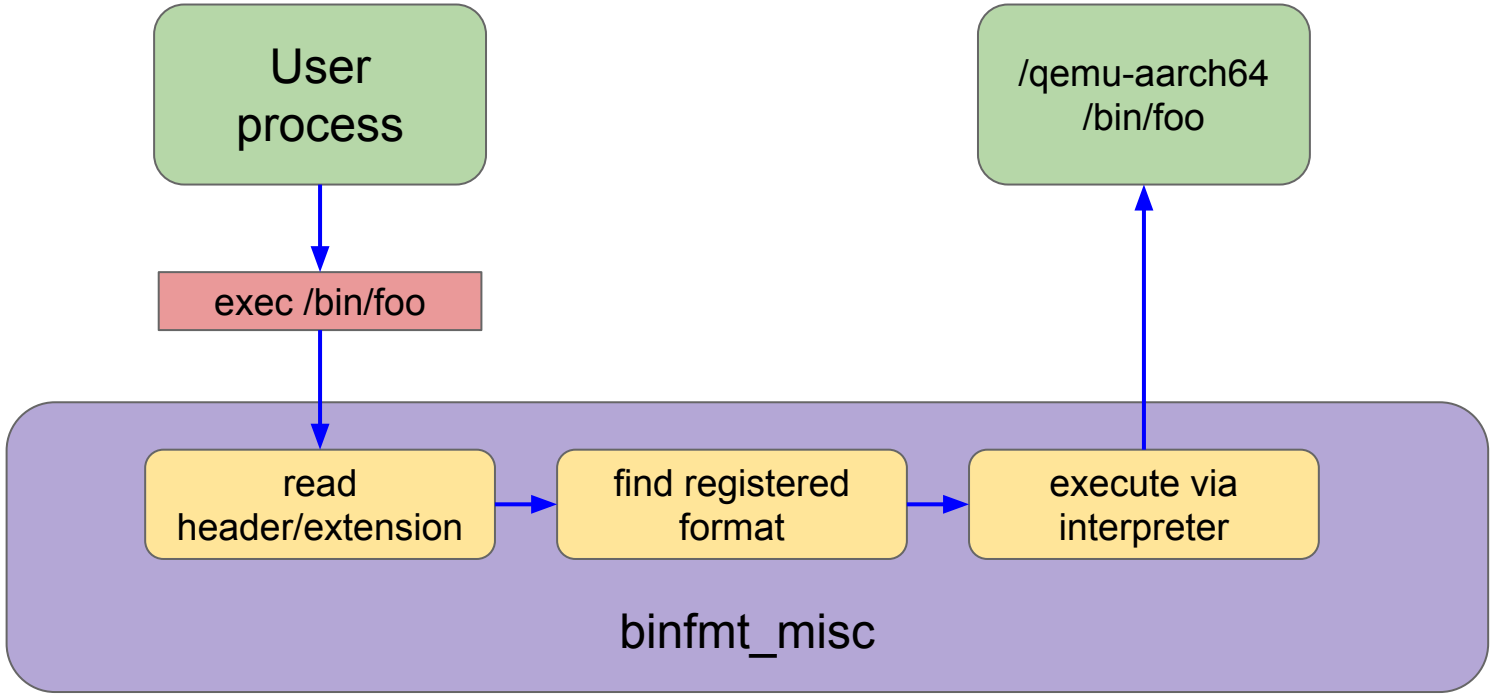
- allows to register custom “interpreters” for specific executables and scripts
- can specify executables by file header or extension

https://www.kernel.org/doc/html/v4.14/admin-guide/binfmt-misc.html

binfmt_misc in action

```
ignat@dev:~$ ./helloarch

-bash: ./helloarch: cannot execute binary file: Exec format error

ignat@dev:~$ echo
":qemu-aarch64:M::\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x02\x00\xb7\x00:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\x
ff\xff\xff\xff\xff\xff\xfe\xff\xff\xff:/usr/bin/qemu-aarch64-stati
c:OC" | sudo tee -a /proc/sys/fs/binfmt_misc/register

:qemu-aarch64:M::\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x02\x00\xb7\x00:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xf
f\xff\xff\xff\xff\xff\xfe\xff\xff\xff:/usr/bin/qemu-aarch64-static
:OC

ignat@dev:~$ ./helloarch

Hello, aarch64!
```

binfmt_misc in action



CLOUDFLARE

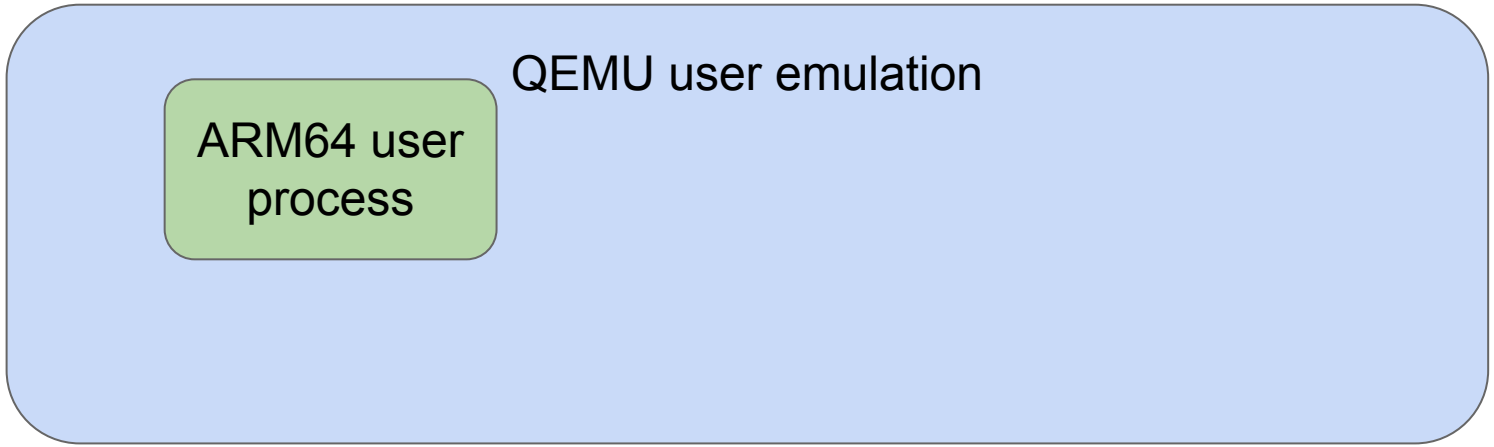
Combining QEMU, binfmt_misc and Docker

```
ignat@dev:~$ cat Dockerfile
FROM arm64v8/debian:stretch
COPY qemu-aarch64-static /usr/bin/qemu-aarch64-static
ignat@dev:~$ docker build -t arm64/stretch .
...
Successfully tagged arm64/stretch:latest
ignat@dev:~$ docker run --rm -it arm64/stretch
root@4e466498353f:/# uname -m
aarch64
```

Foreign arch containers with just qemu

- QEMU translates every system call before passing it on to the kernel

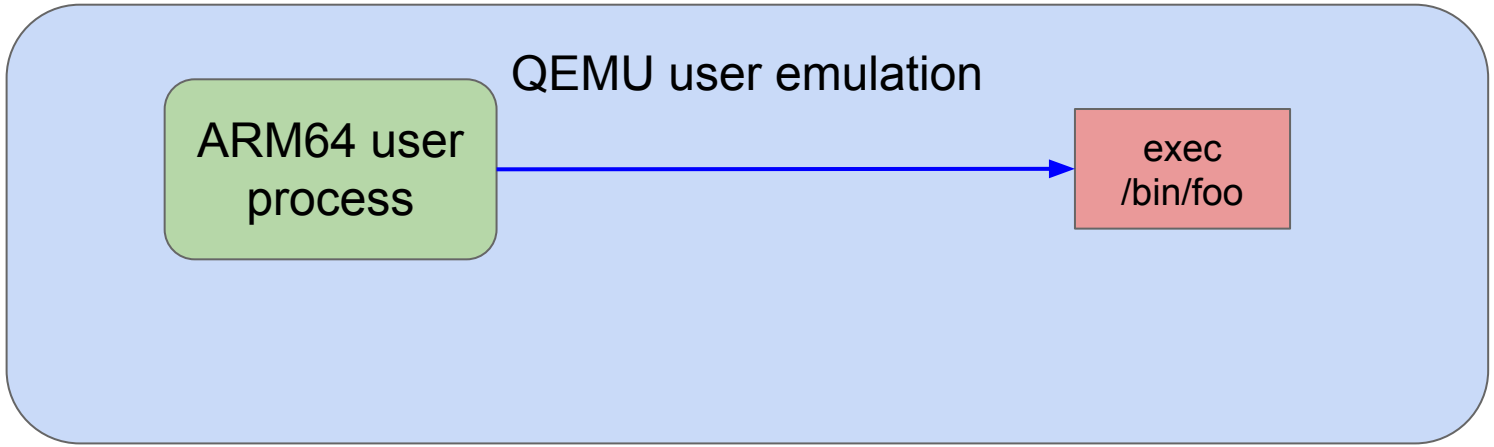
QEMU user emulation



Linux kernel

CLOUDFLARE

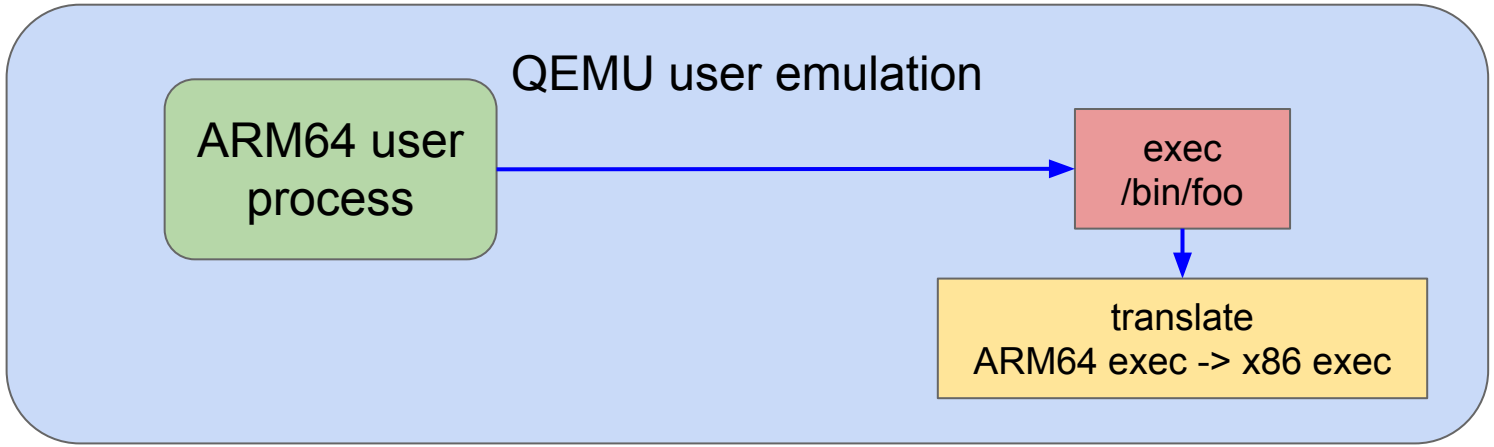
QEMU user emulation



Linux kernel

CLOUDFLARE

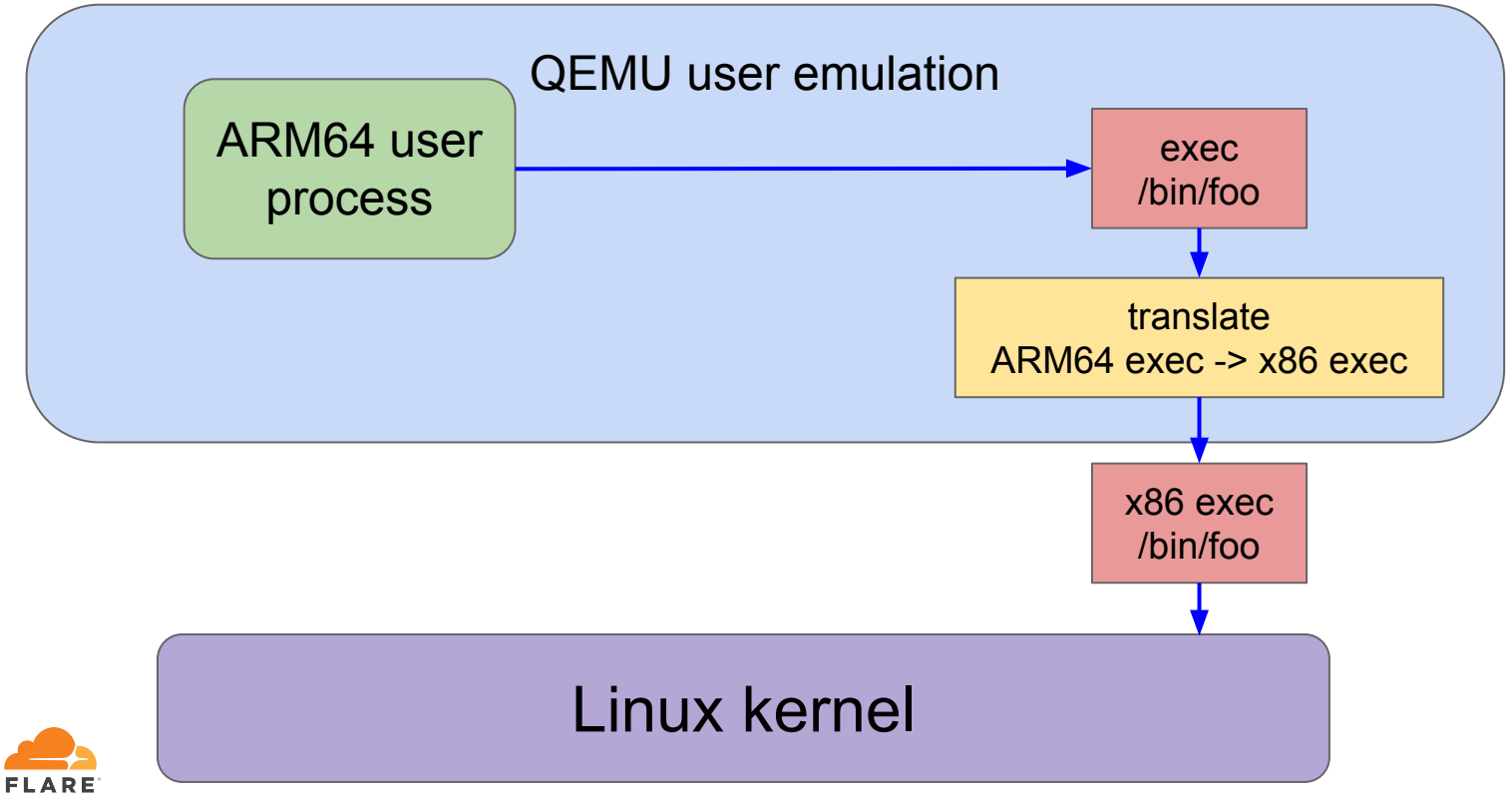
QEMU user emulation



Linux kernel

CLOUDFLARE

QEMU user emulation

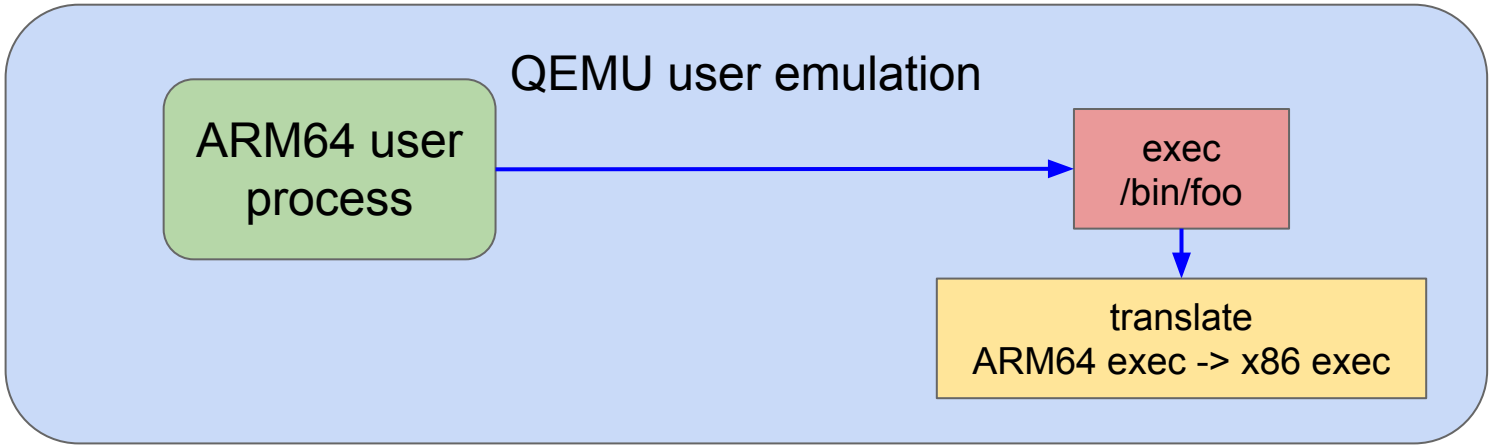


CLOUDFLARE

Foreign arch containers with just qemu

- QEMU translates every system call before passing it on to the kernel
- Why not replicate ***binfmt_misc*** functionality in QEMU itself?

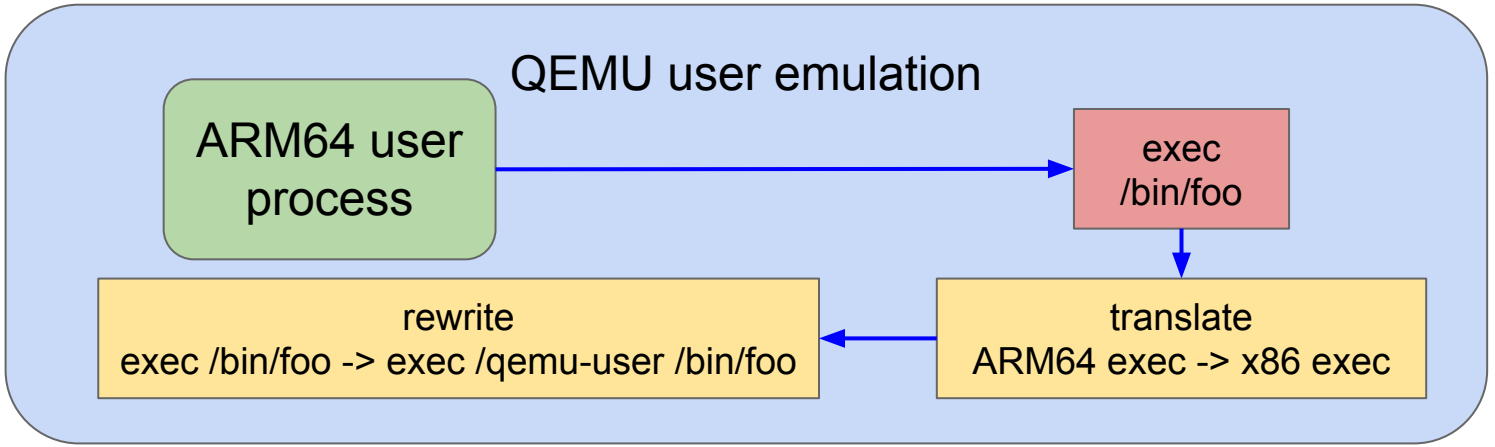
Modified QEMU user emulation



Linux kernel

CLOUDFLARE

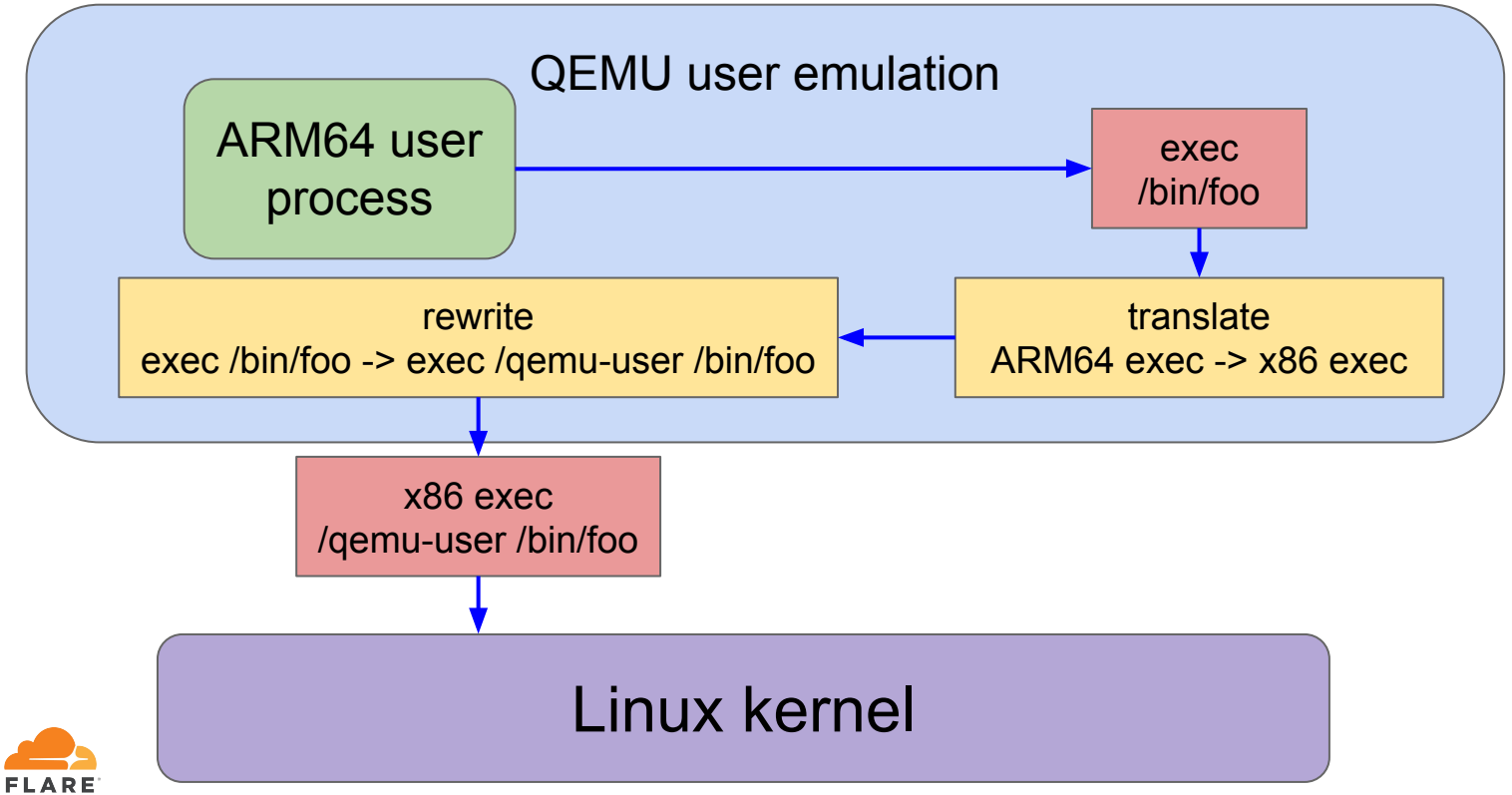
Modified QEMU user emulation



Linux kernel

CLOUDFLARE

Modified QEMU user emulation



CLOUDFLARE

Foreign arch containers with just qemu

- QEMU translates every system call before passing it on to the kernel
- Why not replicate ***binfmt_misc*** functionality in QEMU itself?
- Don’t need to rely on external ***binfmt_misc*** functionality and can create truly self-contained foreign architecture Docker images

Foreign arch containers with just qemu

CLOUDFLARE

Foreign arch containers with just qemu

https://resin.io/blog/building-arm-containers-on-any-x86-machine-even-dockerhub/

CLOUDFLARE

ARM64 Debian Stretch Dockerfile

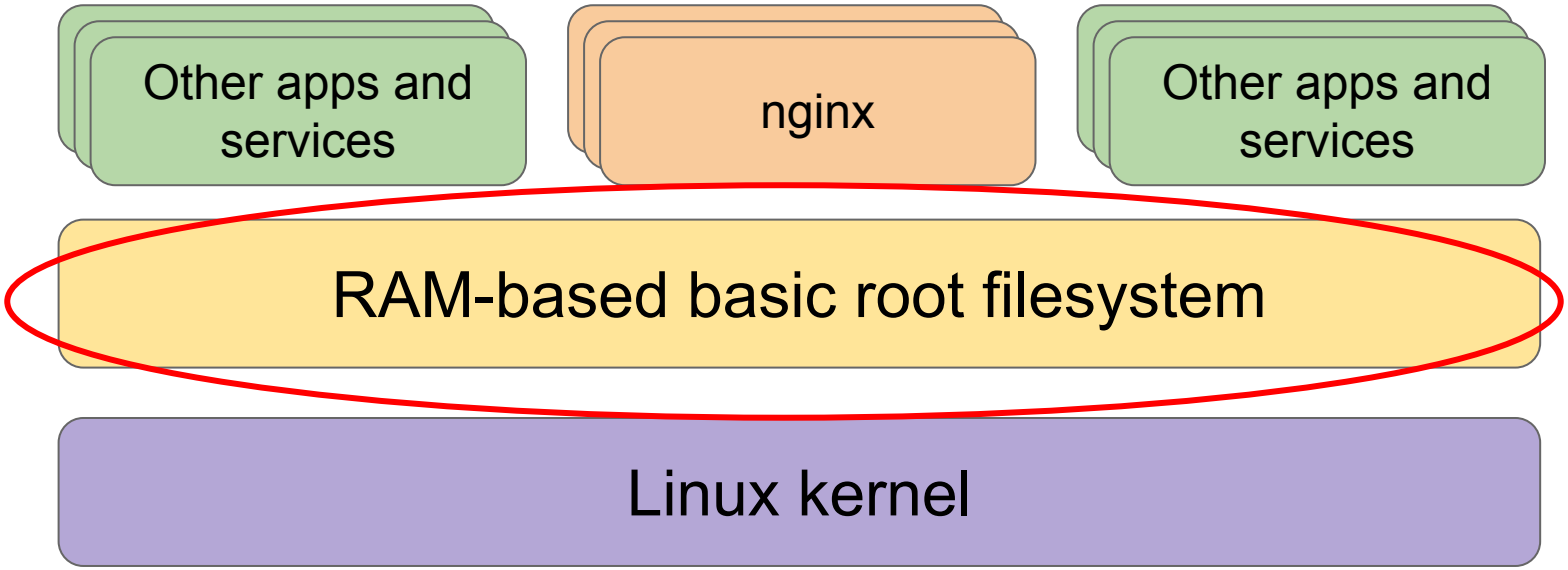
```
FROM debian:stretch-slim as builder

RUN apt-get update && apt-get install -y build-essential python
patch libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev wget

RUN wget https://download.qemu.org/qemu-2.12.0.tar.xz && tar xf
qemu-2.12.0.tar.xz

COPY qemu-execve.patch /qemu-execve.patch

RUN patch -d qemu-2.12.0 -p1 < qemu-execve.patch && \
    mkdir qemu-build && cd qemu-build && \
    /qemu-2.12.0/configure --static
--target-list=aarch64-linux-user --disable-system && \
    make
```

CLOUDFLARE

ARM64 Debian Stretch Dockerfile (cont)

```
FROM arm64v8/debian:stretch-slim

COPY --from=builder /qemu-build/aarch64-linux-user/qemu-aarch64
/qemu-aarch64

SHELL ["/qemu-aarch64", "/bin/sh", "-c"]

RUN apt-get update && apt-get install -y --no-install-recommends
libcap2-bin && \
    setcap cap_setuid,cap_setgid+ep /qemu-aarch64 && \
    apt-get remove --purge -y libcap2-bin && apt-get autoremove -y
&& \
    rm -rf /var/lib/apt/lists/*

ENTRYPOINT ["/qemu-aarch64"]

CMD ["/bin/bash"]
```

Edge server software stack



CLOUDFLARE

ARM64 baseimg

- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs

ARM64 baseimg

- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs
- no changes to the build system

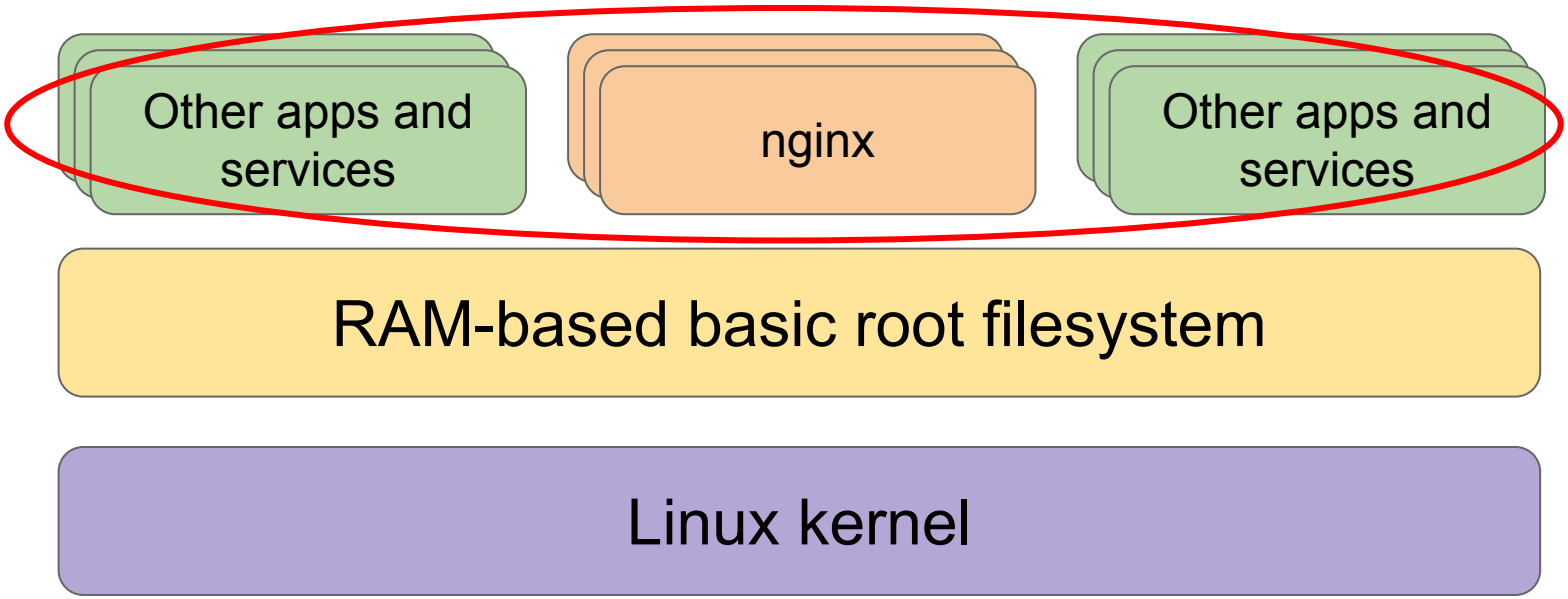
ARM64 baseimg

- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs
- no changes to the build system
- only a single toggle to use ARM64 Debian Stretch as a base instead of x86 one

ARM64 baseimg

- just a minimal Debian image
  - debootstrap --variant=minbase stretch baseimg
  - install config-management agent (salt-minion)
  - package baseimg folder as initramfs
- no changes to the build system
- only a single toggle to use ARM64 Debian Stretch as a base instead of x86 one
- works on x86-based CI out of the box

Edge server software stack



CLOUDFLARE

Porting user-space applications

- reused arm64 Docker based approach
  - reduced avg package porting time from days to minutes
  - no cross-compiling problems, no foreign arch dependency tracing
  - allows devs and CI to run basic tests

Porting user-space applications

- reused arm64 Docker based approach
  - reduced avg package porting time from days to minutes
  - no cross-compiling problems, no foreign arch dependency tracing
  - allows devs and CI to run basic tests
- most of the stack was done in ~1 month

Porting user-space applications

- reused arm64 Docker based approach
  - reduced avg package porting time from days to minutes
  - no cross-compiling problems, no foreign arch dependency tracing
  - allows devs and CI to run basic tests
- most of the stack was done in ~1 month
- although slower build times due to emulation layer
  - ex. nginx takes 2m instead of usual 10s (~10x slower)

Slower build times - it’s a feature!

CLOUDFLARE

Porting timeline

- started porting applications beginning of March

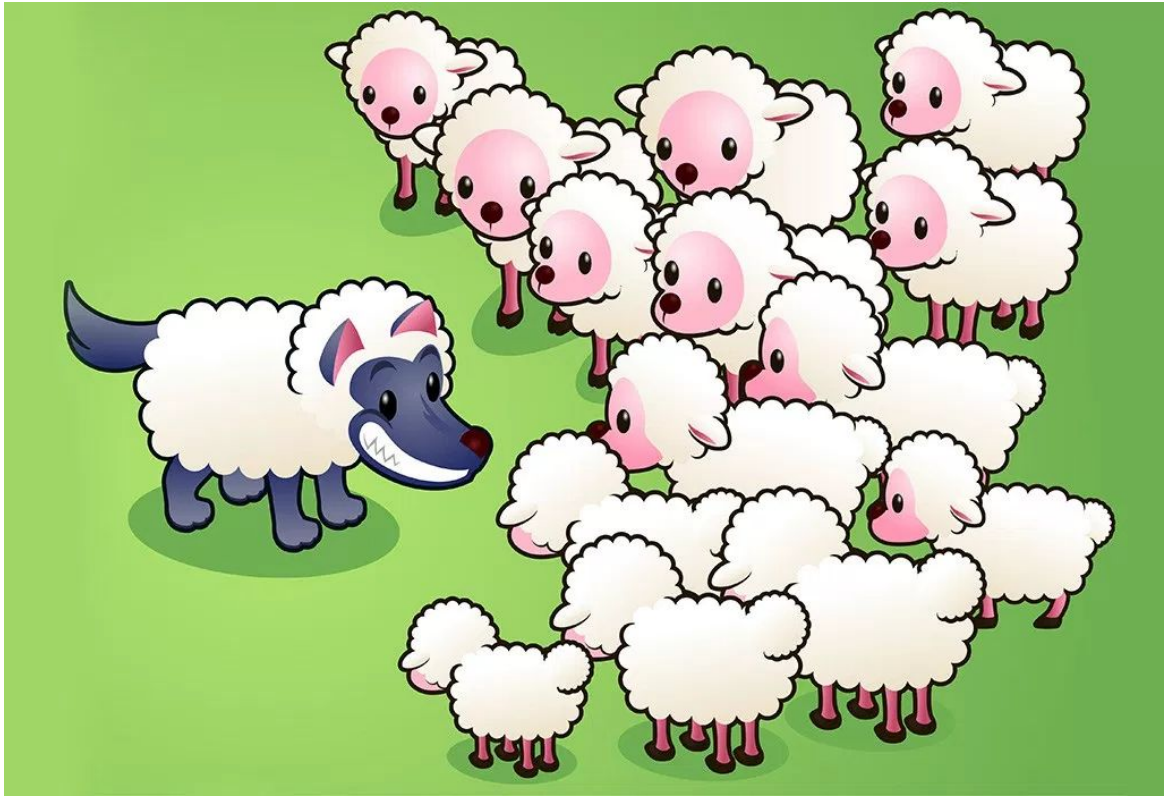
Porting timeline

- started porting applications beginning of March
- March 23rd - served first production DNS request

Porting timeline

- started porting applications beginning of March
- March 23rd - served first production DNS request
- April 18th - served first production HTTPS request with cached content

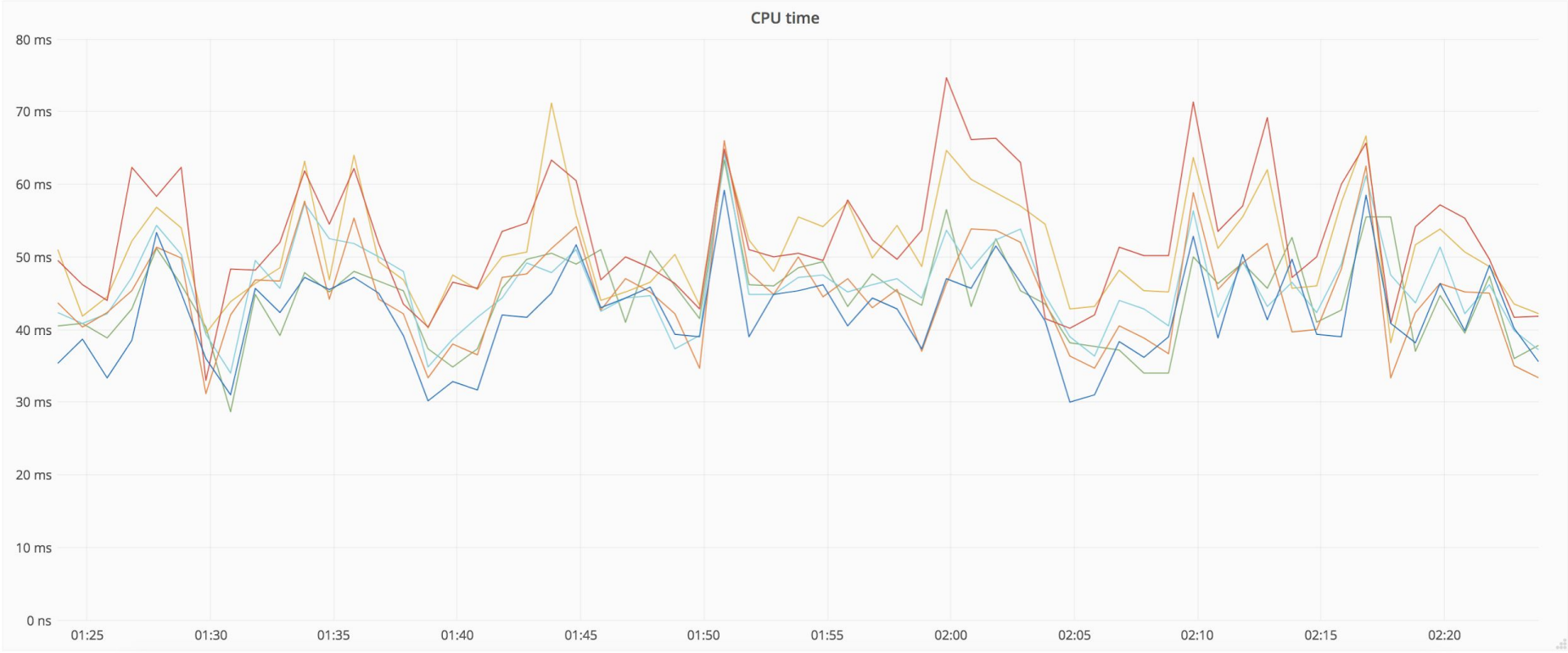
ARM64 in production



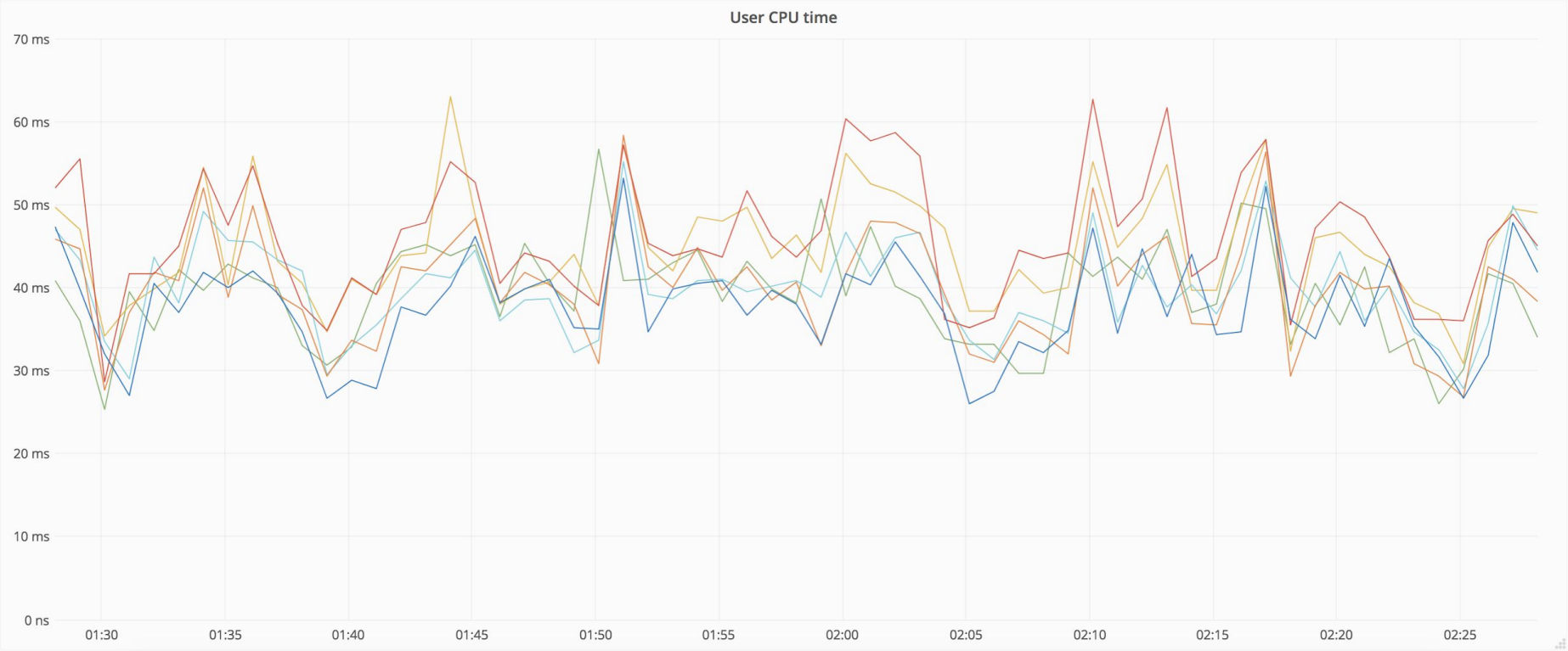
CLOUDFLARE

How is it doing now?

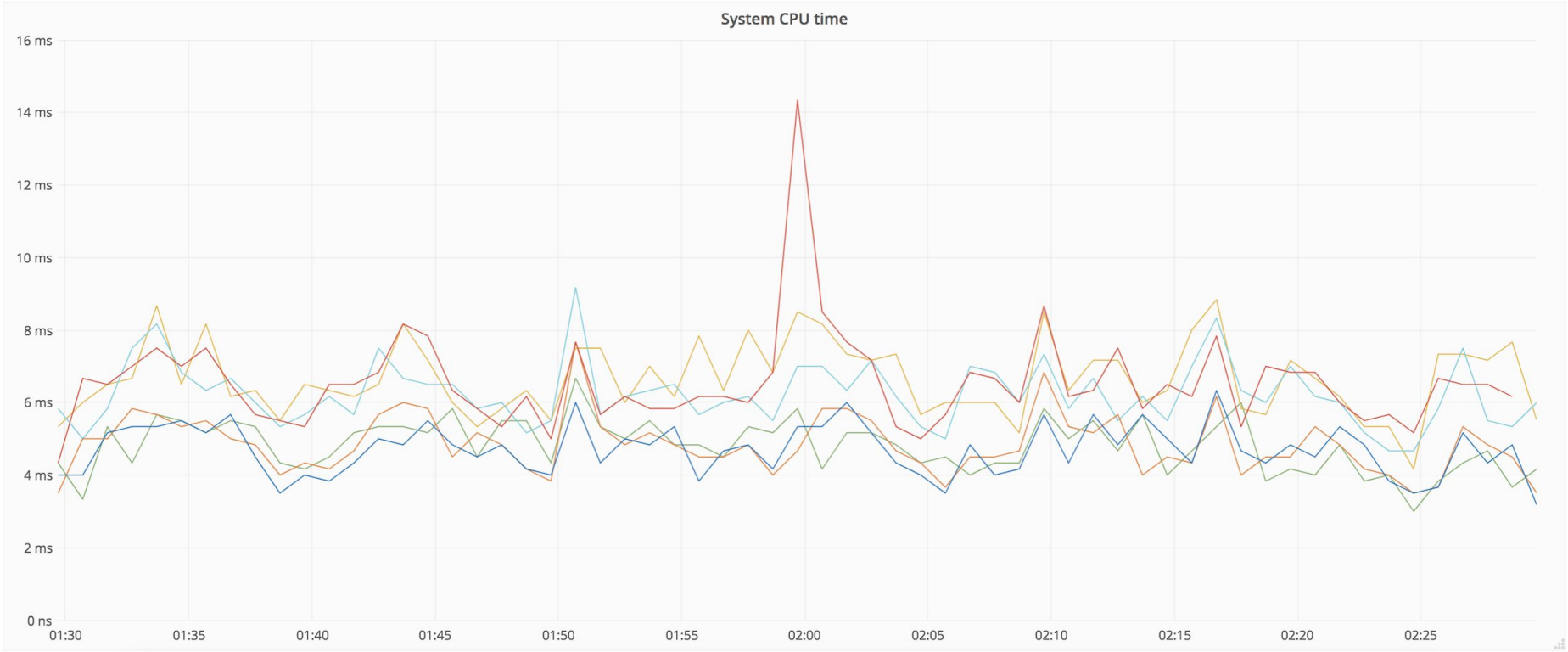
nginx-cache CPU time



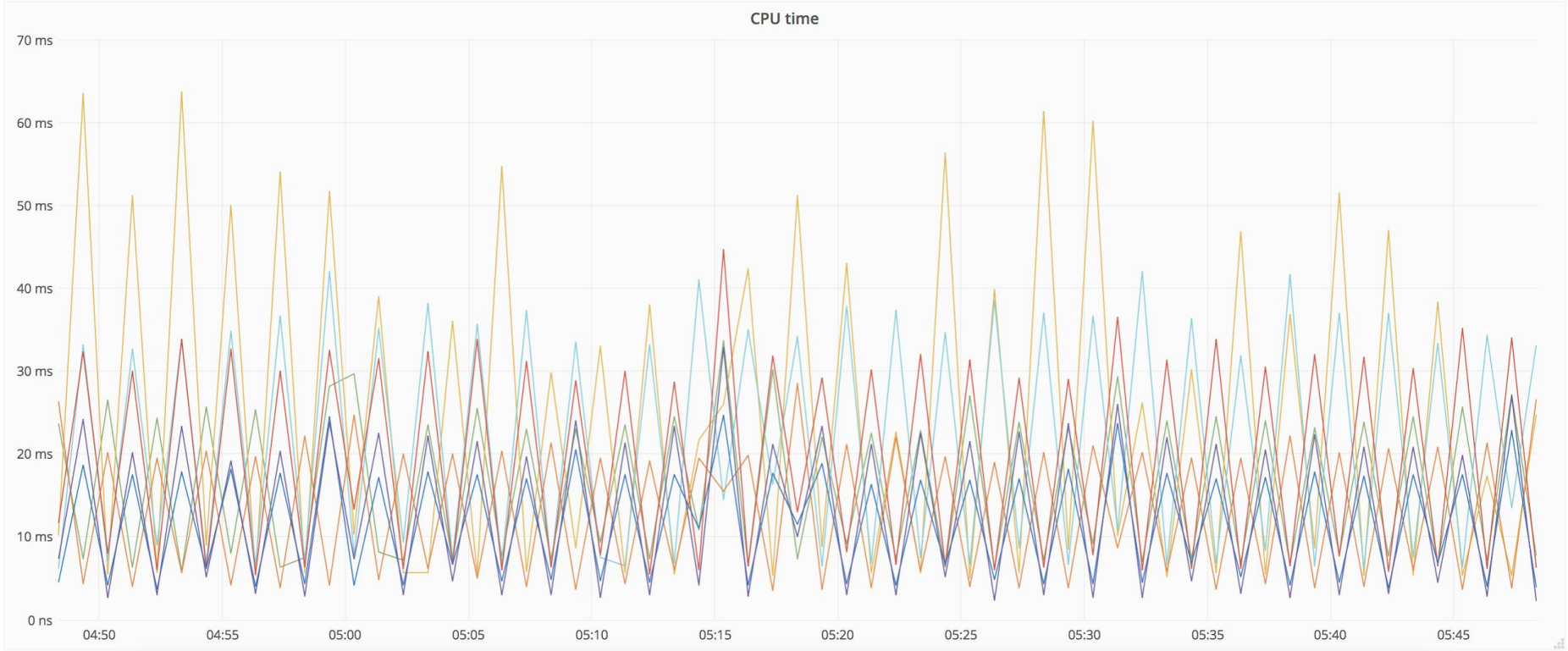
nginx-cache user CPU time



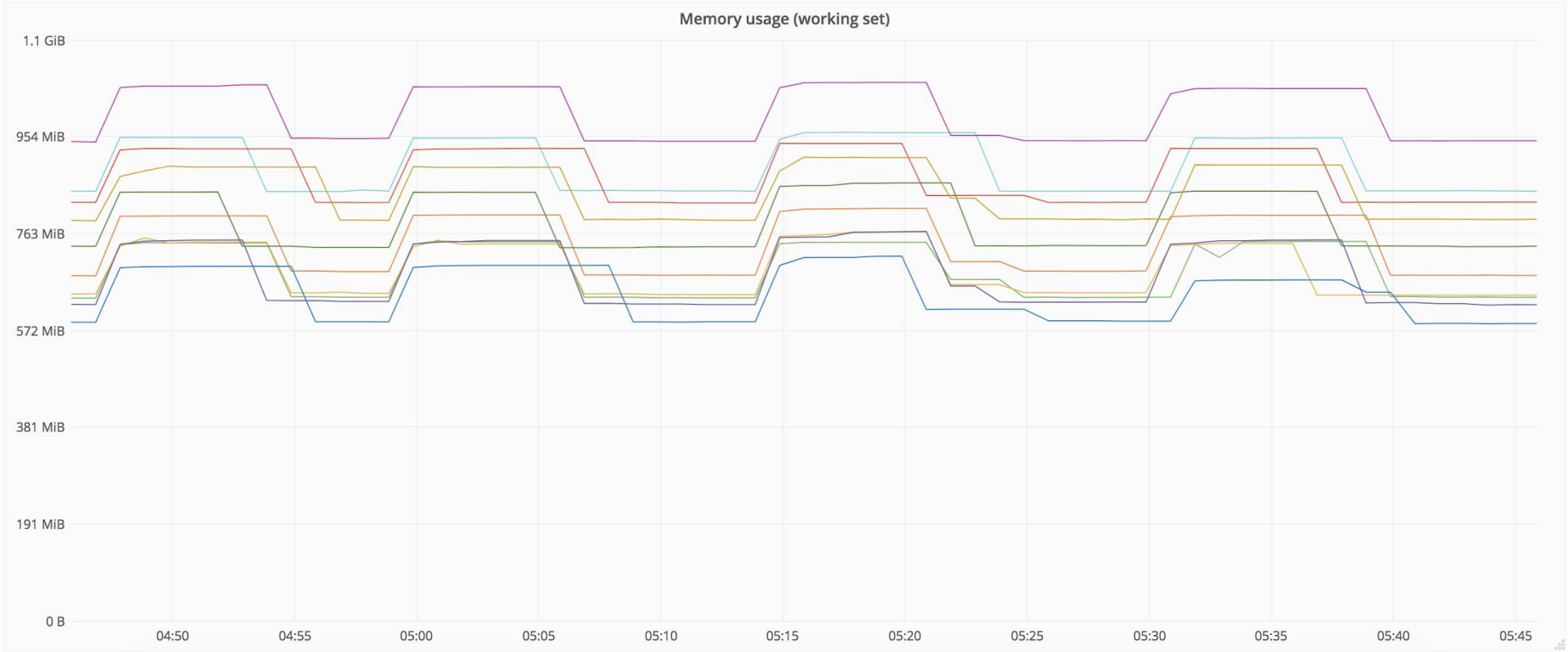
nginx-cache system CPU time



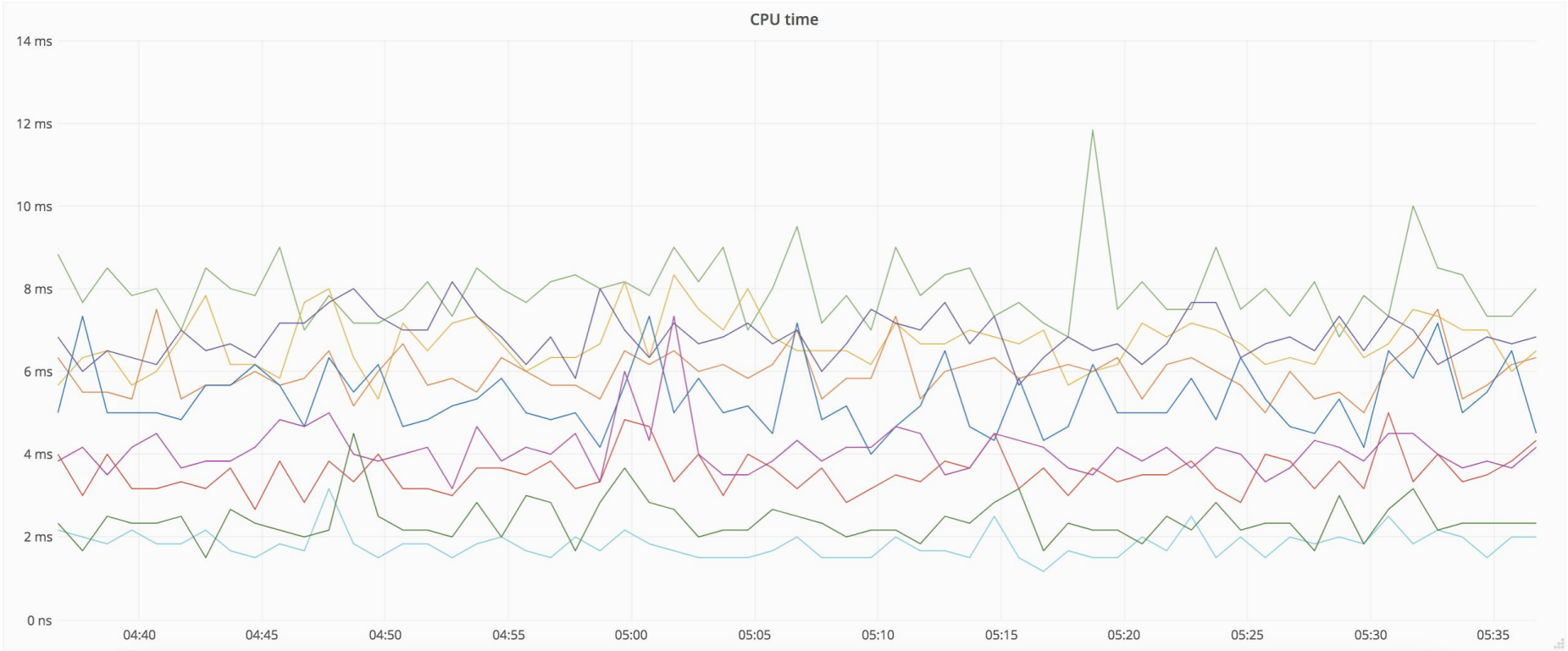
RRDNS CPU time



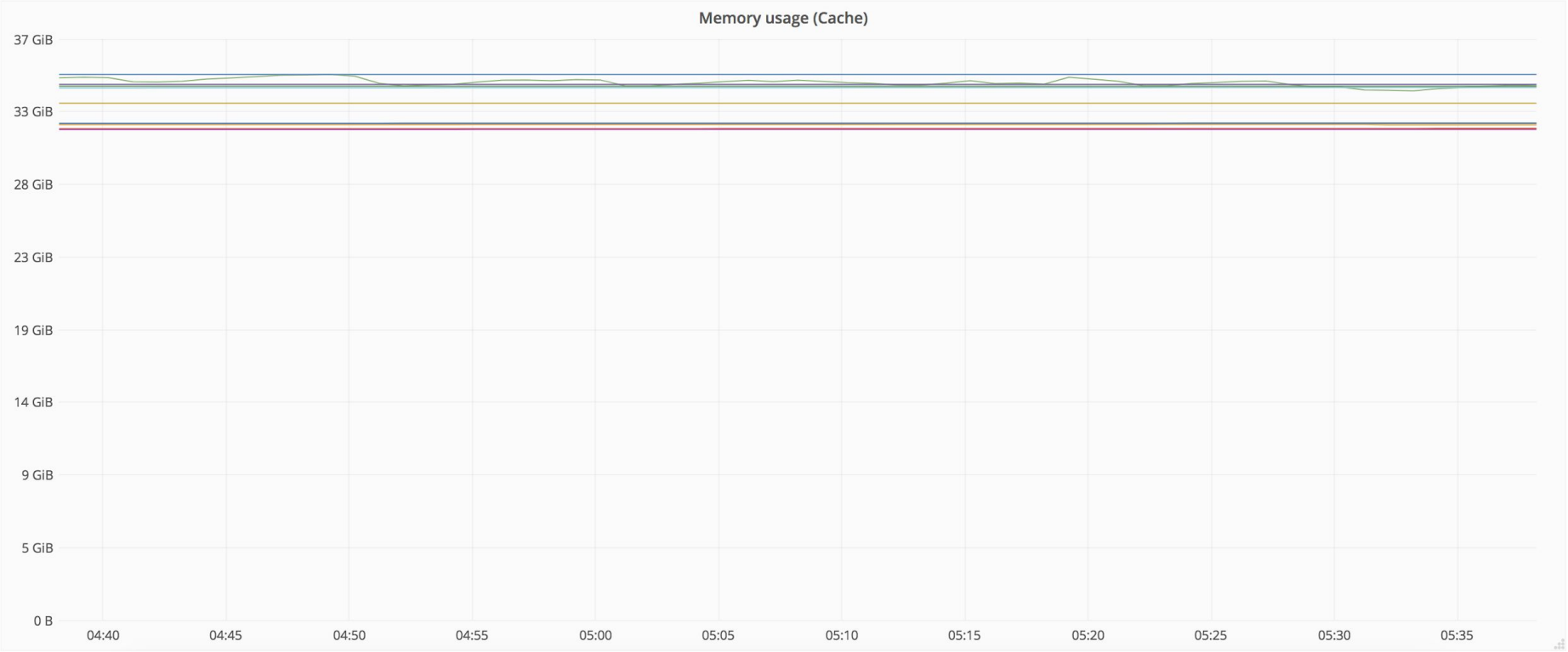
RRDNS memory usage



Quicksilver CPU time



Quicksilver memory cache



Conclusions

- ARM64 is a competitive server architecture
  - most software works OK out of the box
- Migrating to ARM64 is not that hard
  - popular OSes support ARM64 already
- QEMU user emulation + Docker provides a low-cost quick-start solution to port in-house software to ARM64
  - in both hw costs and effort cost
  - minimal disruption to the dev process

Some links

- https://blog.cloudflare.com/arm-takes-wing/
- https://blog.cloudflare.com/neon-is-the-new-black/
- https://blog.cloudflare.com/using-go-as-a-scripting-language-in-linux/
- https://www.kernel.org/doc/html/latest/admin-guide/binfmt-misc.html
- https://resin.io/blog/building-arm-containers-on-any-x86-machine-even-dockerhub/

Thank you!