Toolchain Strategy For Arm Server CPUs

Siddhesh Poyarekar
The Next 20 Minutes Of Our Lives

- Embedded vs Non-Embedded Universes
- Toolchain Project Schedules
- Distribution Schedules (Fedora and Ubuntu and RHEL, oh my!)
- Planning CPU support
Who Am I?

- Developer Services Technical Lead
- GNU C Library Maintainer, gcc, binutils-gdb committer
- Fedora and RHEL package maintainer
- Hack on a variety of FOSS projects, toolchain or otherwise
Board Support Package

- Documentation
- Cross-Compiler Toolkit (GNU based or LLVM based)
- Firmware Image (zephyr, freertos, etc.)
- Debugging Utilities (gdb, openocd, etc.)
The Embedded Universe

- Everything available from a single place
- Users expect hardware vendor to provide software
  - No separate expectations from the software
  - Everything should just tie in together
- Hardware vendor has full control
  - On how to release the software
  - On when to release the software
- Upstreaming of support not very important
  - Middle end or Armv8-specific changes through Arm or Linaro
  - Minimal backend-only changes to support their target hardware
  - Don’t have to deal with community acceptance for quirky hacks
The Non-Embedded Universe

BOARD

Distributions
- Fedora
- Debian
- Ubuntu
- RHEL
- Suse
- Gentoo *
  ...

Projects
- Gcc
- Glibc
- Binutils
- Linux
  ...

* In solidarity with the 5 remaining Gentoo users
The Non-Embedded Universe

- Laptops, Desktops, Workstations, Servers, Cloud
- Complex software ecosystem, unique requirements
  - Support cycles of up to a decade
  - Insane Rigorous backward compatibility requirements
- So Software is it’s own universe, nay, TWO UNIVERSES!
- Users rarely interact with the Project world they only know distributions
- Hardware vendors have limited control
  - Need to ensure distributions support their hardware
  - Distributions take software only from upstream. Downstream maintenance is already too expensive!
  - Upstreaming is critical, upstream timing even more so
  - Balance IP confidentiality and product availability timing
- It’s GNU toolchain all the way, at least for the next 10 years
Upstream

- All projects have their own release schedules
- Gcc: Annual releases, around April every year
  - Stage 1 freeze in November - no major features after this
  - Stage 3 freeze (there’s no stage 2!) in January - only critical bug fixes after this
  - Stage 4 freeze == Release time
- Glibc: Two releases a year, February and August
  - Development freeze 1 month prior to release
  - You better watch out: Christmas holidays just before the freeze*
- Binutils: Two releases a year, January/February and July/August

* If you’re lucky, the release manager may be from India and may be working through Christmas.
Upstream: Vendor Targets

- **Gcc**
  - Upstream support for microarchitecture by January Stage 3 freeze
  - Safer target: November Stage 1 freeze
  - Hard targets: Missing them will lead to a delay of a year
  - Distributions don’t backport new -mcpu flags
  - Distributions are very wary of compiler backports

- **Glibc**
  - Optimised routines: earlier the better
  - Backport policy conservative but...
  - Backports typically well contained, so can go back a few releases
  - Plan early, release later

- **Binutils**
  - Often ignored, but some hacks can be beneficial
  - Distribution backport policy is conservative
  - Add the cpu flag, don’t wait till you find out that it’s too late
Distributions

- Fedora, Ubuntu, etc.
  - Six month release cadence
  - Support lifecycle of 2 years

- RHEL, Ubuntu LTS
  - Long release cadence. RHEL-8 is coming out after over 5 years!
  - Conservative backport policy, gets tougher as releases progress
  - 10+ year support lifecycle with half yearly update releases
  - Fedora feeds into RHEL, Ubuntu feeds into LTS
Distributions: Vendor Strategy

- Fedora, Ubuntu, etc.
  - Earlier the better

- RHEL, Ubuntu LTS
  - Major releases are the only entry point for gcc
  - Update releases may backport glibc optimisations but gets harder with time
  - gcc and glibc from 1-2 years ago from the date of major release
  - Backports very conservative, so get your flags and tuning scaffolding in early
But Wait! There’s More!

- Distributions have Developer Toolchain Packages
  - Red Hat Developer Toolset is an example
  - More recent toolchains, refreshed every 2-3 years
  - A good target for gcc optimisations
  - Typically use gcc from the previous year
  - Not a good target for glibc - core distribution is the only option
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org