Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023

Linux Plumbers Conference | Richmond, VA | Nov. 13-15, 2023

kernel: build system outputs and workflows (and how to balance them)

Bruce Ashfield, AMD

The kernel as part of a build system

Can’t keep everyone happy at the same time!

… but yet we try!

The foundation and pillars …

images, boot components,
tools, utilities, containers/VMs …

kernel, modules, dtbs, userspace configuration, firmware, debug, SBoM, CVE …

config

reprod
ucibility

flexibility

maintenance
&
support

image
creation
&
build
speed

debug

versions, release cadence, support duration

source (vendor, upstream, BSP), patches, configuration, tools …

The ins and outs (high level) ...

- ins:
  - source
  - configuration & policy
  - security (keys, etc)
- outs:
  - kernel and supporting binaries
  - boot artifacts (scripts,device tree,firmware)
  - packages and images
  - traceability / licensing / dbug / SBoM / CVE

The extras ...

- Tightly coupled components
  - compiler / libc headers
  - tools (lttng, perf, systemtap .. etc)
- SDK / build artifacts
  - shared kernel source
- Containers, VMs, unikernels
- Out of tree modules, depmod

Personas / Workflows ...

- Release / loadbuild / production
- Developer: kernel, userspace
  - build -> debug
- Integrator
- Distributor
- Community member / Contributor

Is there a primary persona / target ?

The kernel in Open Embedded

- Flexible provider model (virtual/kernel)
  - source, patches, configuration, etc
  - Presents challenges (many versions, different support, varied source / patch, tools)
- Multiple output types
  - kernel (multiple formats: simple or complex)
  - initramfs, images
  - signed, unsigned
  - kernel modules are separately packaged

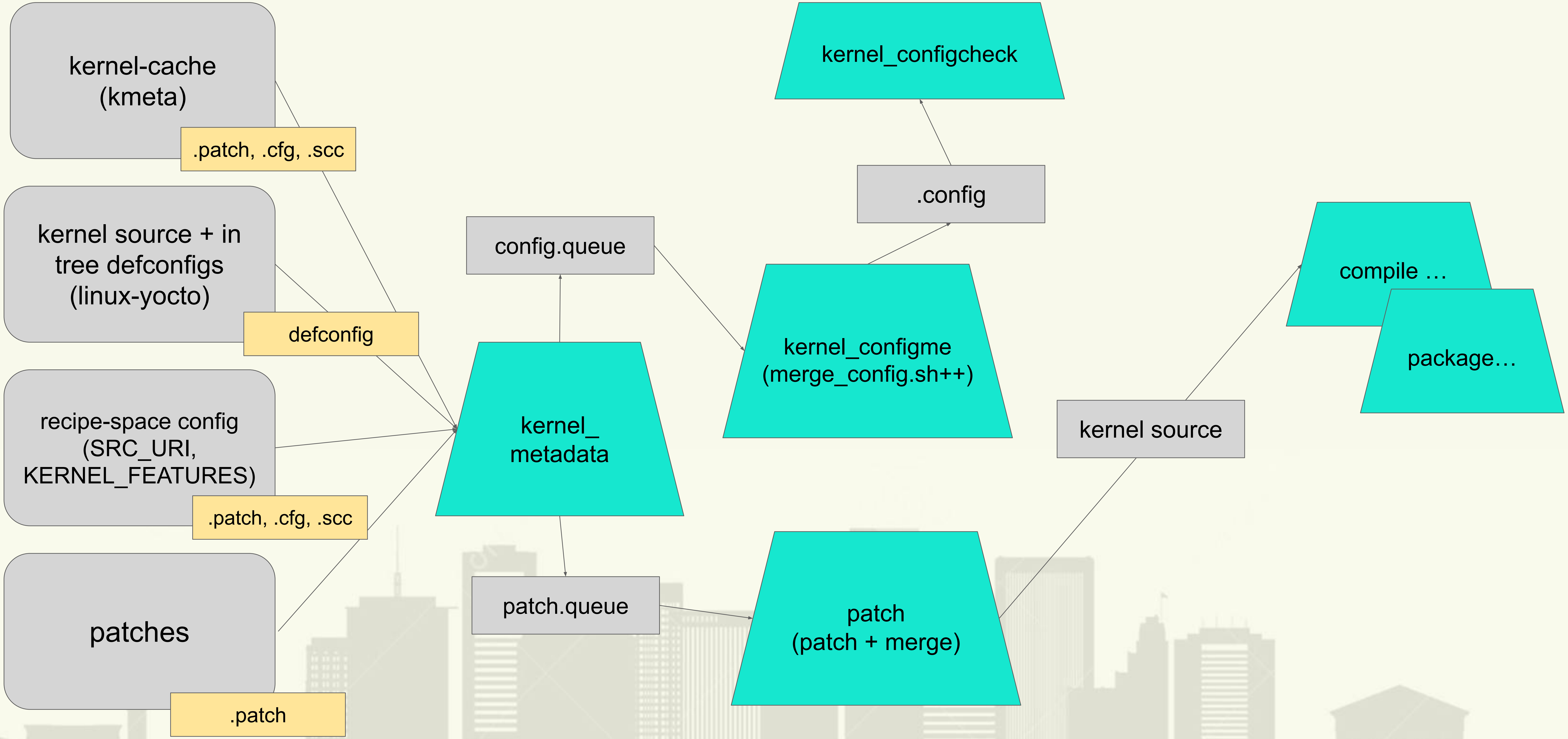
What about a reference ?

The Yocto / OE reference kernel

- Release cadence and explicit version testing
- Drives kernel workflows
    - bitbake/OE core support
- Launching point for production / commercial offerings
- Vertical / specific configurations testing
    - -rt, -tiny, -standard, developer, k.org
- Configuration / extension model
- Collection point of contributed BSPs
- Support the validation / testing of the ecosystem: tightly coupled packages, uapi, libc, containers, etc

Find and fix the ‘hard to solve problems’

The reference kernel: build flow



Challenges (open questions!) ...

- (infinite) different entrenched workflows ...
  - configure, build, deploy, boot, debug
  - patch and source management ..
- Many different trees (it's a forest!)
  - hundreds of BSPs .. how to EOL?
  - version expansion!
- Inconsistent quality and testing
- Support / security updates

More Challenges ...

- Many different ways the kernel can be consumed
    - SDK ? binary ?
    - where can the kernel be rebuilt ?
    - reference only or production ready ?
- Which use cases to optimize ?
    - Is build performance important ?
- New requirements: rust ...
- Are tools provided ?
- What is “standard” packaging ?

Thoughts (not answers!) ...

- Offer workflows, but don't mandate them
  - Includes source management
  - Almost any overhead is "too much"
  - Those that want to adopt it ... will
- Provide flexibility, but focus testing on a reference
  - i.e: embedded, enterprise, hobby .. etc)
- Do no prematurely optimize a use case
- Provide a reference to gather momentum / resist fragmentation
  - Document!! (support model, lifespan, updates, etc)

OE kernel .. what's next ...

- Enhanced testing (we've found some unique issues)
    - More kernel specific on-target testing
        - kselftest, stress tests, etc
    - stress testing
    - additional kernel type testing (-rt, -dev)
- New Architectures
- Binary Reference Kernels
- Expand boot testing coverage
    - more hardware targets
    - more image types
- Streamline developer workflows
- Performance tracking

Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023