How to choose a kernel to ship with a product

Willy Tarreau
HAProxy Technologies
Kernel Recipes 2015

Outline

- Target
  - People who don’t use distro kernels
  - People who run their own distros
  - People who don’t know they’re currently at risk
- Identify your needs
- What not to do
- Remaining options
- How to improve your situation

Background
Past and present activities

- First kernel patch around 1995 on kernel 1.2.13
- Maintainer of very old kernels (2.4, 2.6.32, ...)
- Been shipping remotely managed gateways at Exosec from 2002 to 2008
- Been shipping HAProxy load balancers since 2006

Challenges

=> Maintain systems up to date WRT bugs/vulns:

- Gateways: operate remotely without local help
- LBs: avoid regressions casting a bad image of the product and company in general

What absolutely needs updates ?

Few critical components on a system :

“The ones allowing you to fix a bug and then to fix the failed attempt to fix this bug”

In short (YMMV) :

- Kernel
- Libc
- sshd

Not all of them in fact

- Libc very rarely needs to be updated in a product's life
- Most products don't depend on SSHd, pick the last one and you're OK.

Kernel is special

The kernel :

- provides drivers for **your** device
- provides drivers for other obscure devices
- provides various network protocols
- provides all the security layers

=> ***suffers orders of magnitude more bugs than any other component found in such an appliance***.

Kernel is sensitive!

A missed kernel upgrade in field can lead to :

- System malfunction / instability / outages
- Data corruption / data loss
- Local and remote DoS
- Local root

=> ***angry customers, bad image, loss of time and money***

Kernel is too sensitive!

In case of bad or failed upgrade in field :

- Have to drive there (managed gateways)
- angry customer + RMA (LBs)

=> Common question :

“should I risk an upgrade or risk a bug?”

No such thing as validation in lab

The Linux kernel is a monster project.

- Current size around **19.5M** lines
- Linux 4.2 added around **1M** lines of code (**5%**)

=> Nobody can verify there's no regression.

At best we can be certain that **there are** regs...

...and you don't want them in your product!

More numbers

Kernel 2.6.32 was released on 2009/12. Patches merged from 2009/12 to 2012/05 :

3325

Patches merged from 2012/05 to 2015/09 :

724

Uncertain fixes dropped since 2012/05 :

~3000

=> ***The kernel is a living thing !***

Facing a difficult choice

- How long will the product live ?
- What initial level of stability is expected ?
- Does the product accept hardware extensions in field ?
- Will it be possible to update in field ?
- Will the customers see what is running ?
- Will the customers expect updates ?
- Is downtime affordable ?
- Are security issues really a problem ?

Stick to what works

- Don't change a kernel branch in the middle of a product's life cycle if breakage is not an option.
- Backport the fixes you need only.
- Stable kernels are made for you.

=> ***OK but which one ?***

How the kernel quality evolves

- All fixes are first merged into mainline before -stable
- -stable lasts 2-3 months, + about 2 years for LTS.
- **Typos / thinkos** are found after a few days / weeks

  => *they will be fixed in **-stable***
- More **subtle bugs** can take months/years to be detected

  => *only **LTS** will have a chance to fix them*
- **Architectural shortcomings** require massive rework

  => *only a **version upgrade** will bring them*

First corollary on kernel quality

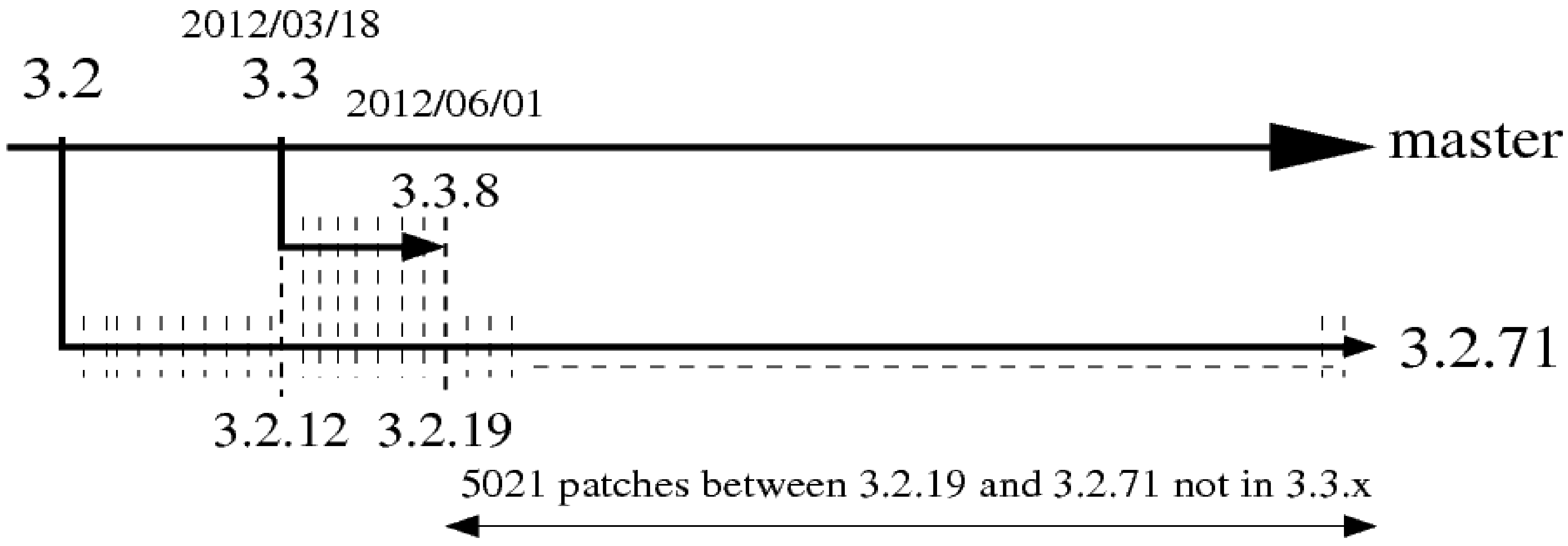
Claim: ***Unmaintained EOLed kernels contain a lot of subtle bugs that are hard to detect.***

You don't believe me ? OK, let's see :

First corollary on kernel quality

Claim: *Unmaintained EOLed kernels contain a lot of subtle bugs that are hard to detect.*

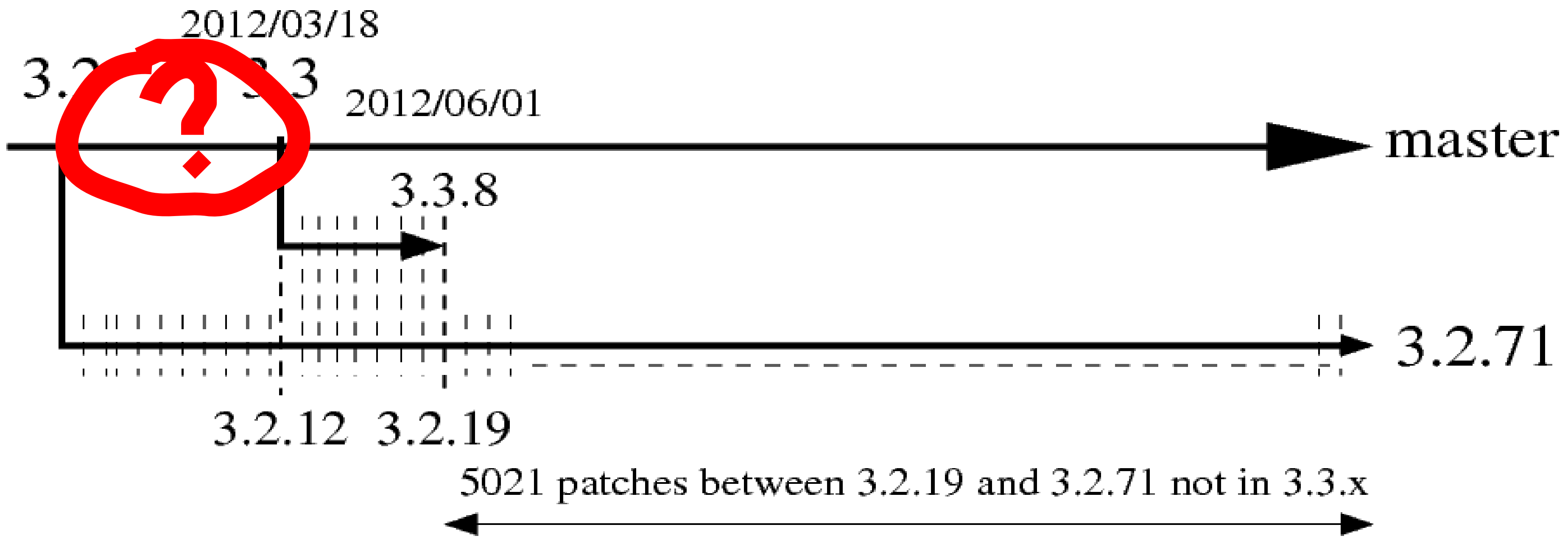
You don't believe me ? OK, let's see :



First corollary on kernel quality

Claim: *Unmaintained EOLed kernels contain a lot of subtle bugs that are hard to detect.*

You don't believe me ? OK, let's see :



Immediate conclusion

NEVER EVER USE EOLed KERNELS!!!

If you ***really*** need them, help with their maintenance and try to get your work supported by more recent ones.

So what should you do ?

- Pick a kernel that lasts **at least as long as** your product's life (or major version's life)
- **Degree of confidence** should match your customers' expectations (ADSL router, load balancer, file server, power plant controller, ...)
- Backport missing features (arbitration needed)
- But avoid backporting infrastructure changes!

You're not alone!

- All distro vendors and product vendors are in the same boat.
- Use your **distro's kernel** if it matches your needs, **it's free**.
- Otherwise join forces to maintain a specific kernel for as long as you need.

=> ***But please, oh please, avoid needless effort duplication caused by adding more LTS versions.***

How to participate

- Upstream your local fixes and improvements (*but not your local crap, thank you*)
- Test stable kernels and report good/bad status
- Scrutinize changelogs to check for failed backports
- **Maintain, publish, and advertise** a stable branch if not possible otherwise

Don't maintain your own kernel!

Out-of-tree patches = pain. Need to rebase, apply at the wrong place, etc... Sometimes patches are bogus.

<u>Speaker's experience :</u>

- Got bored with local patches in Feb 2005 (pre-Git era), started the public “hotfix” tree for 2.4 kernels (2.4.29+)
- Most of them were adopted by maintainer Marcelo Tosatti
- People started to use them, proposing more fixes, sending feedback about some of them being wrong or not working as expected, leading to newer versions
- Eventually all of them got merged or had an equivalent fix.

  => *first “stable” tree without this name :-)*

Don't maintain your own kernel(2)

- Been shipping 2.4 in our products till 2011 (EOS 2013)

  => *Very stable, but amount of efforts was becoming too high to support new hardware (x86_64) and features (IPv6)*
- Moved to 2.6.27 and dropped many patches
- Then 2.6.32, dropped more patches, and replaced some

  => *met a handful of bugs that were already fixed upstream*
- And then 3.10 and dropped more patches

  => *one regression caused by a local patch after a -stable update*.
- Now 3.14 and kept very few patches

  => **less out-of-tree patches = less work and less bugs**

4 types of stable kernels now

Quest for kernel quality has led to 4 types of stable kernels :

- Official kernel.org **stable** kernels by Greg KH, started with 2.6.11 by Greg&Chris.
- Official kernel.org **LTS** kernels by Greg, started with 2.6.27 and initial attempts with 2.6.20, 2.6.22, and 2.6.25.
- **Extended** LTS kernels maintained by individuals after Greg, started with 2.6.16 by Adrian, replicated with 2.6.20, 2.6.27, 2.6.32, 2.6.34, 2.6.35, 3.4.
- Kernels picked by **distro maintainers** : 3.2, 3.12, 3.18

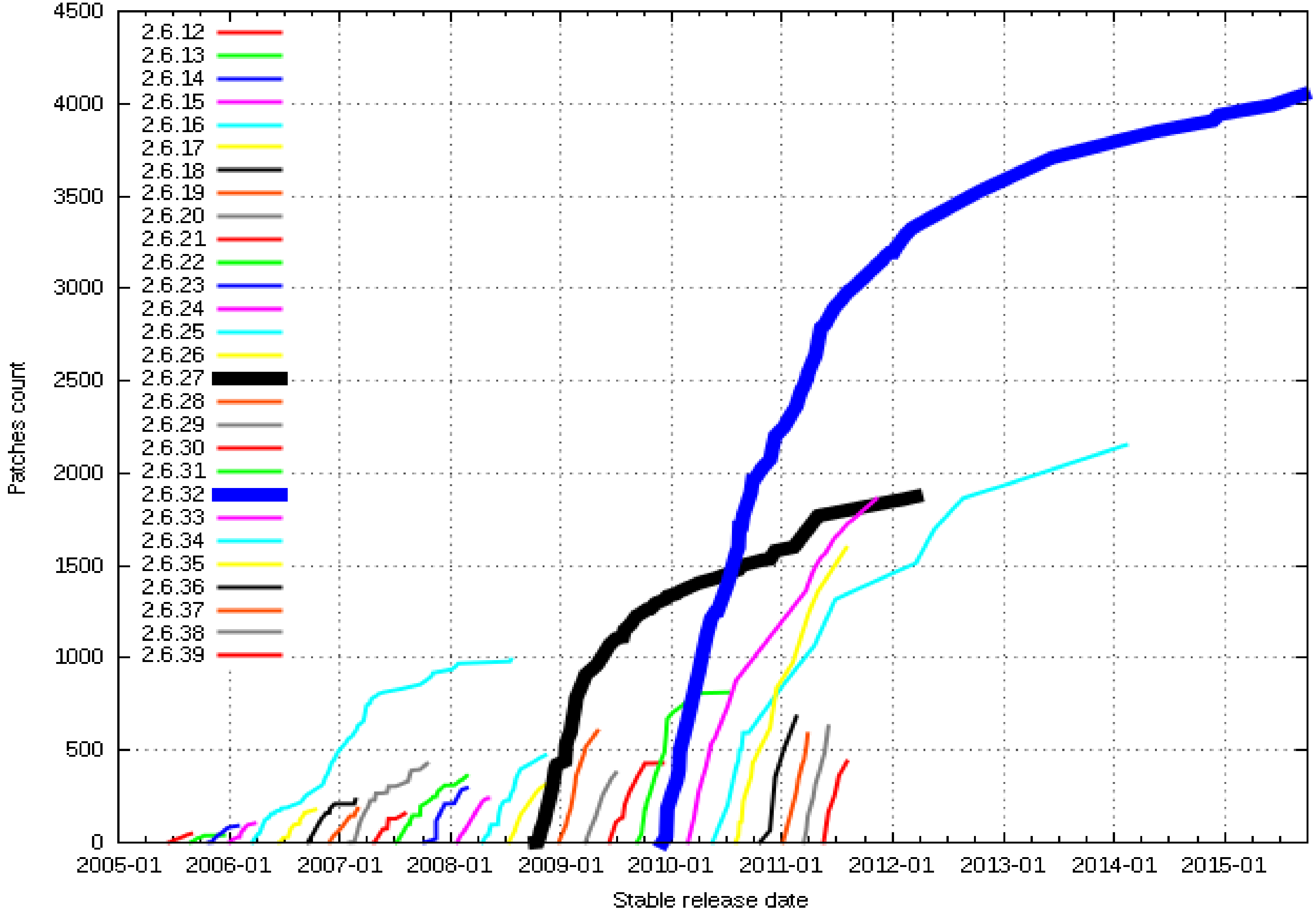
=> **8-9 stable branches in parallel now including 7 LTS**

See https://kernel.org/category/releases

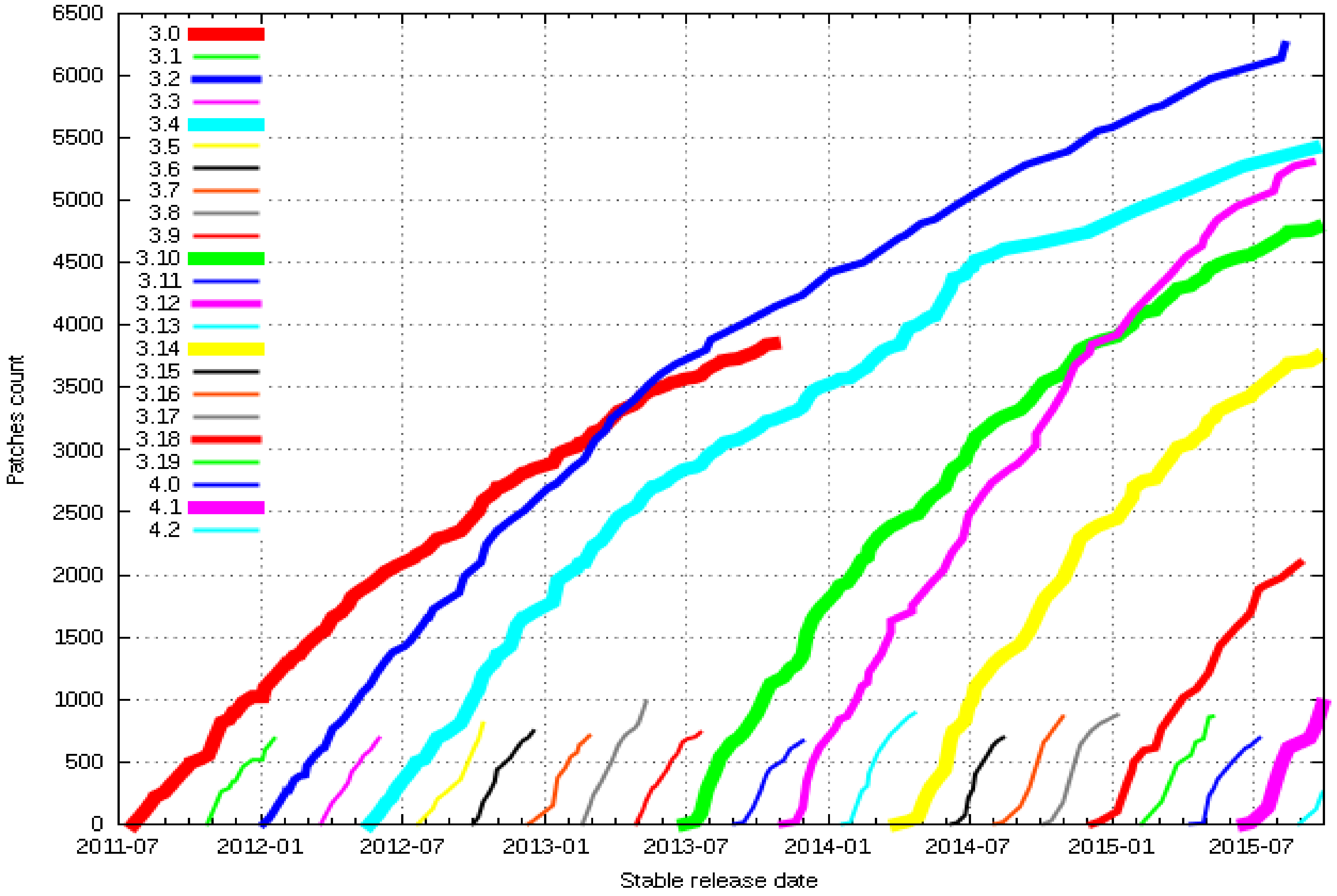
How to choose ?

- 7 LTS are a lot, but they offer a lot of choice.
- Do they all evolve at the same speed ?
- Are they all maintained similarly ?
- Let's see...

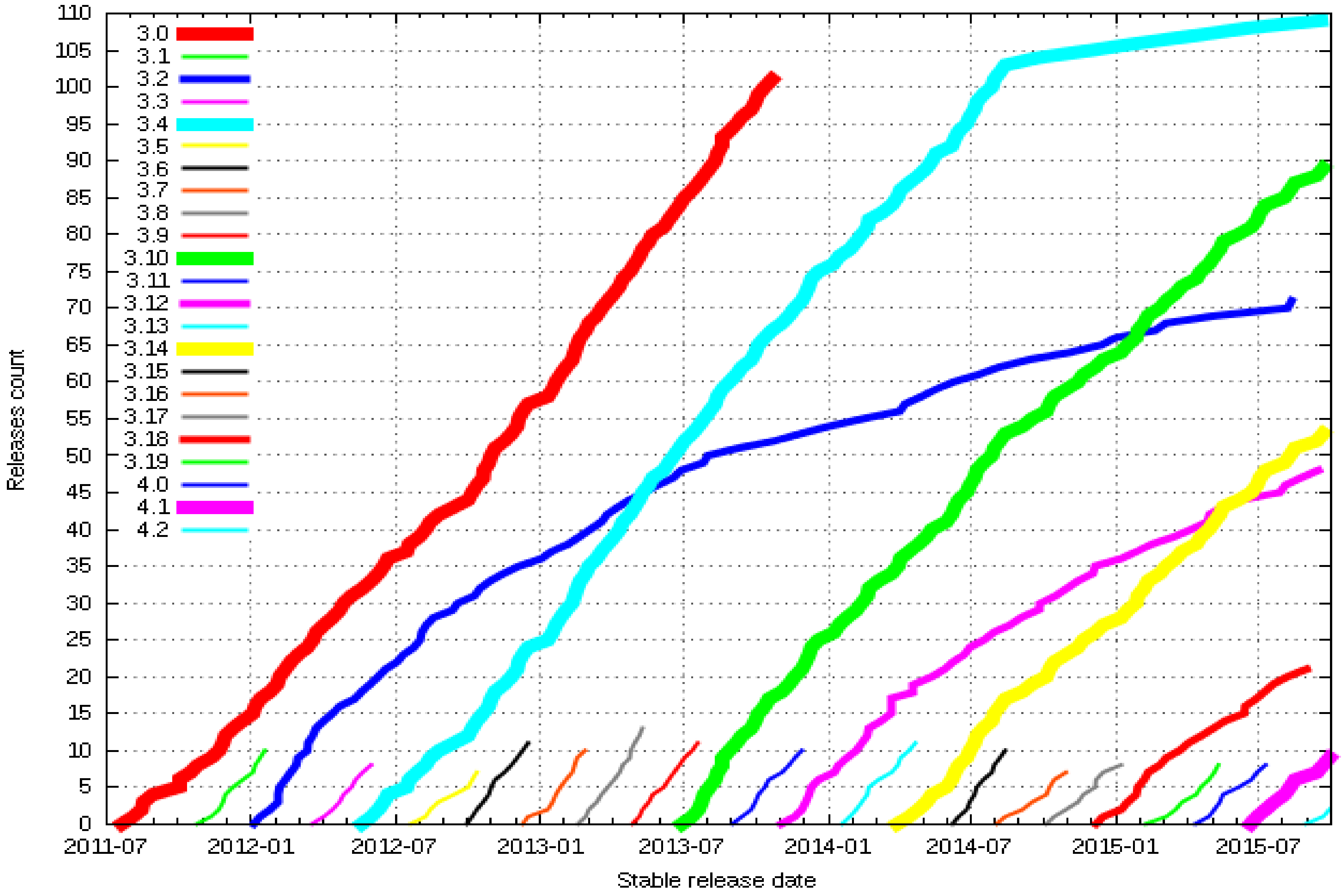
How 2.6 kernels evolved



How 3.x/4.x kernels evolve



Different release pacing



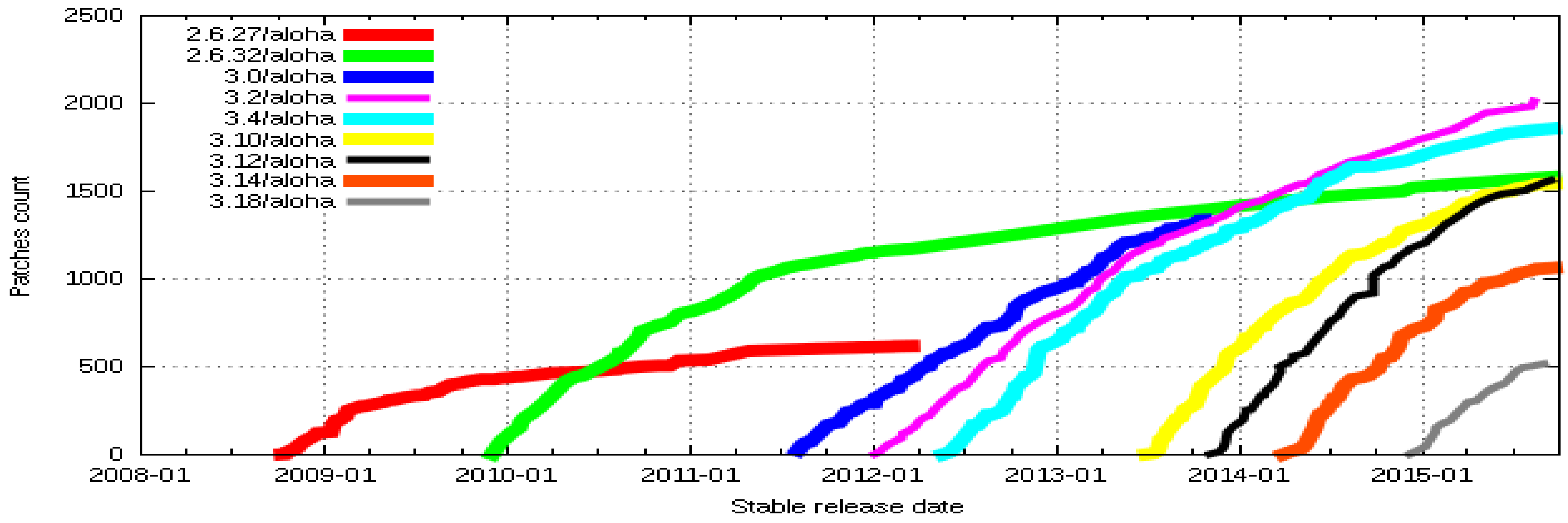
Analysis

- Greg's kernels are very regular, about 1 version per week.
- Distro kernels merge more patches but space their releases more over time.
- Maybe distros backport more features for their users that are less of interest for your products.

=> **don't be fooled by numbers, focus only on the kernel areas relevant to your product**

Patches per area of interest

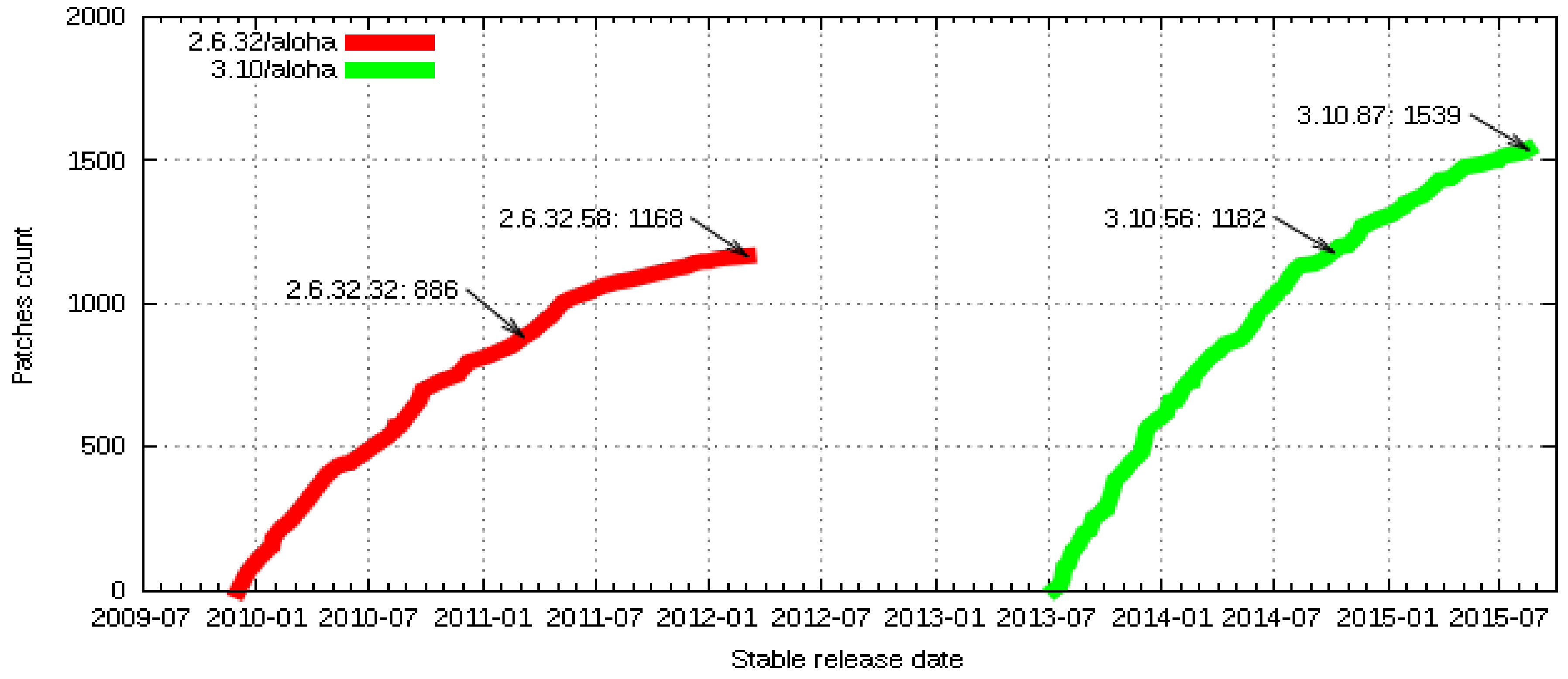
The ALOHA Load balancer is mostly intested in **kernel, mm, ipc, lib, net, drivers/net, arch/x86 :**



=> 3.x are pretty much all the same finally...

… but look closer!

LTS quickly stabilizes



=> 3/4 of fixes Greg takes are merged in the first 15 months!

Try to limit the effort

- **newer kernel** = less work but more care and risks
- **older kernel** = less risks but more work
- 3/4 of patches in 15 months is very appealing, that leaves **one year of free maintenance** for LTS kernels after we start shipping.
- Our products ship a new major version every year, with 3 years of support

  => **2 years of maintenance left for us.**
- 3.2 and 3.4 would have saved one extra year, but we needed more recent features.

=> **We picked 3.10 in 2014, updated to 3.14 in 2015, maybe 4.1 in 2016.**

Don't pick too old kernels

Very old kernels are not the cheapest ones to maintain :

- Standard fixes get painful after 2 years
- Security fixes are very hard to backport (doc replaced some fixes in 2.4)
- Developers are not much willing to help when kernels are too old
- Less reviewers and testers for each new release

  => **regressions start to increase again past some point** (eg: 2.6.32)

Distro kernels getting really good

- Less frequent but maintained longer (4-5 years)
- Strong experience within these teams
- Most are now part of the official kernel.org trees.

=> Definitely a good choice when compatible

<u>One regret:</u> too bad they don't stick to the regular LTS versions and share maintenance efforts. Debian's kernel team has been very helpful on 2.6.32, they could have picked 3.4 instead of 3.2 and avoid duplicating efforts. Same regarding 3.16.

Post-shipping: be careful

- Always review kernel changelogs

Hint: look at -stable only, and it's linear!

- **Don't upgrade if there's nothing for you, especially in July-August!**
- Very rare -stable regs but happen once in a while and can also be caused by your own patches
- Test. Your code could rely on a bug that was recently fixed!
- Report anything suspicious as soon as possible while it's still fresh in developers' mind.

Plan for next major upgrade

- Try to identify what next version you'll have to upgrade to
- Try to identify if someone is willing to maintain it for you or does another one you're fine with.
- Start to run it on your laptops then servers long before you ship it. *One more reason for distro kernels!*
- Test as often as possible and report breakage

That's all folks!

Thanks!

Questions / Comments / Jokes ?

Contact: Willy Tarreau <willy@haproxy.com>

HAProxy is hiring talented people. Contact us!