Unification of embedded CPU variants

HISAO MUNAKATA
RENESAS SOLUTIONS CORP
hisao.munakata.vt(at)renesas.com

Disclaimer

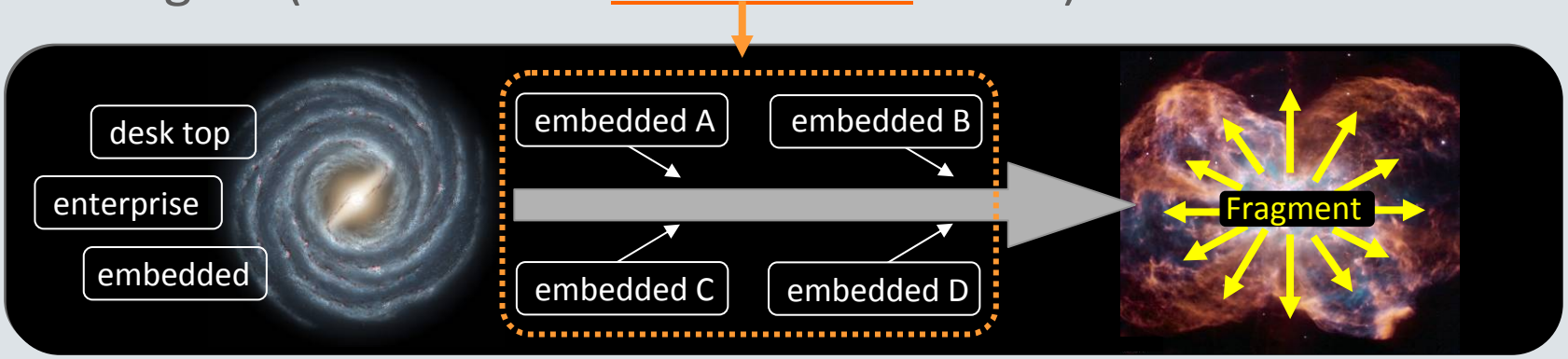
Everything I say here is just my opinion and not the opinion of my employer Renesas. As our experience of embedded architecture other than SuperH are quite limited, my understanding here might not be correct. If you have objection to my opinion, please collect me at any time. I appreciate your permissiveness.

Agenda

- Is Linux getting fragmented due to embedded device support ?
  - Device driver complexity ------ fat “defconfig” problem
  - Distribution fork ---------- Android kernel problem
  - CPU support confusion --------- vendor tree problem
- Lesson learned and our observation
  - SuperH kernel support (as architecture provider)
  - ARM kernel support (as ARM Linux newbie)
- Future direction of embedded Linux
  - How should we manage diversion of embedded Linux
  - Possible tactics

Anxiety of future embedded Linux

- Now that Linux becomes primary OS for network aware consumer and/or industry products - what people call “embedded device” - we all depend on embedded Linux, however...
- Contribution for kernel development from embedded world is still in low gear (well-known contribution issue)



- Recently I have noticed that excessive diversity of embedded device potentially break current Linux kernel harmonization. This is my problem recognition for this talk.

example : ARM defconfig complexity

http://lwn.net/Articles/391372/

ARM and defconfig files

By **Jake Edge**
June 16, 2010

The kernel tree for the ARM architecture is large and fairly complicated. Because of the large number of ARM system-on-chip (SoC) variants, as well as different versions of the ARM CPU itself, there is something of a combinatorial explosion occurring in the architecture tree. That, in turn, led to something of an explosion from Linus Torvalds as he is getting tired of "pointless churn" in the tree.

A pull request from Daniel Walker for some updates to `arch/arm/mach-msm` was the proximate cause of Torvalds's unhappiness, but it goes deeper than that. He responded to Walker's request, by pointing out a problem he sees with ARM:

> There's something wrong with ARM development. The amount of pure noise in the patches is incredibly annoying. Right now, ARM is already (despite me not reacting to some of the flood) 55% of all arch/ changes since 2.6.34, and it's all pointless churn in
>
> ```
> arch/arm/configs/
> arch/arm/mach-xyz
> arch/arm/plat-blah
> ```
>
> and at a certain point in the merge window I simply could not find it in me to care about it any more.

He goes on to note that the majority of the diffs are "mind-deadening" because they aren't sensibly readable by humans. He further analyzes the problem by comparing the sizes of the x86 and ARM trees, with the latter being some 800K lines of "code"—roughly three times the size of x86. Of that, 200K lines are default config (i.e. defconfig) files for 170+ different SoCs. To Torvalds, those files are "pure garbage".

example : ARM defconfig complexity (cont.)

http://lwn.net/Articles/391372/

In fact, he is "actually considering just getting rid of all the 'defconfig' files entirely". Each of those files represents the configuration choices someone made when building a kernel for a specific ARM SoC, but keeping them around is just a waste, he said:

> And I suspect that it really is best to just remove the existing defconfig files. People can see them in the history to pick up what the heck they did, but no way will any sane model ever look even _remotely_ like them, so they really aren't a useful basis for going forward.

Another problem that Torvalds identified is the proliferation of platform-specific drivers, which could very likely be combined into shared drivers in the `drivers/` tree or coalesced in other ways. Basically, "we need somebody who cares, and doesn't just mindlessly aggregate all the crud". Ben Dooks agreed that there is a problem, but that "many of the big company players have yet to really see the necessity" of combining drivers. He also noted that at least some of the defconfig files were being used in automated build testing, but did agree that there are older defconfigs that should be culled.

Dooks also had a longer description of the problems that ARM maintainers have in trying to support so many different SoCs, while also trying to reduce the size and complexity of the sub-architecture trees. Essentially, the maintainers are swamped and "until it hits these big companies in the pocket it [is] very difficult to get them to actually pay" for cleaning up the ARM tree and keeping it clean in the future.

Linus claiming about ARM kernel (defconfig /driver) unregulated complexity

Characteristics of embedded

- Uniqueness of each hardware (no standard abstraction)
  - BIOS not used in embedded device
  - Non standard peripheral controller (USB, LAN, Graphics,..)
  - Unique on-chip controller (many variant in even same vendor)

Example referred

- bloated “defconfig” to support various embedded platform.
- bloated device driver entry to support non-standard devices

- Embedded CPU variant support
- CPU instruction set incompatibility
- Chip specific implementations

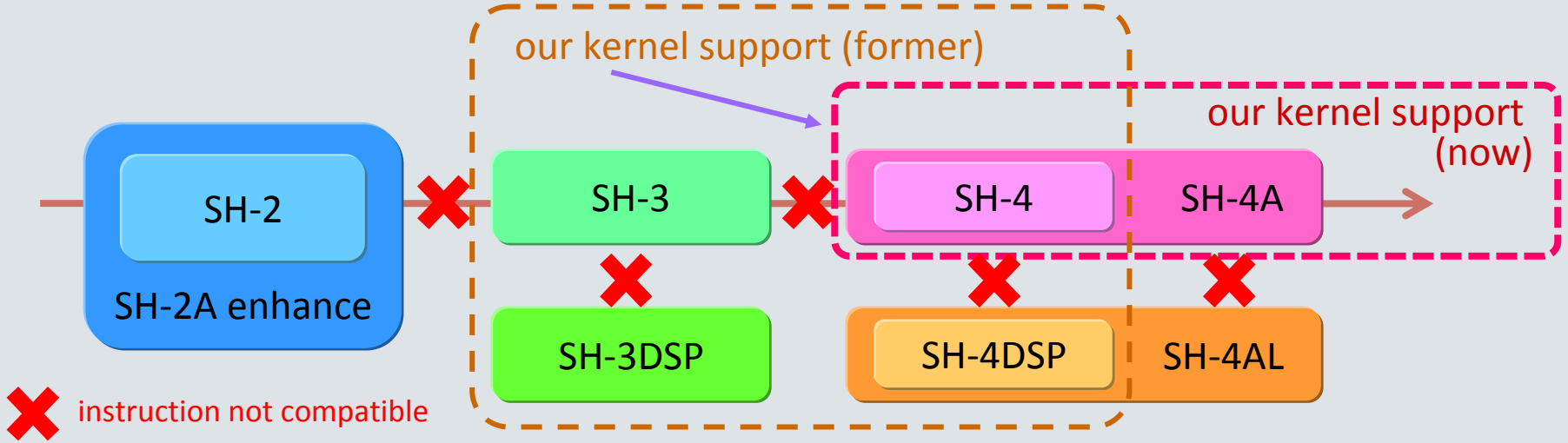
CPU flavor : Intel IA32

- Intel IA32 : **Clean and consistent**
  - Intel defines IA32 instruction set as common infrastructure
  - There are some enhanced instruction set to utilize new core capability, but still IA32 instruction can run on any Intel CPU including latest 64bit architecture CPU.
  - Linux kernel and distribution support IA32 as baseline
  - Some additional architecture can be supplied as a option
  - Most of IA32 device are from Intel
  - Thus IA32 is default target architecture in Linux world, and x86-64 instruction support in now integrated to one kernel.

All Intel CPU can run IA32 as common base instruction

CPU flavor : SuperH

- Renesas SuperH : Bit fragmented → Getting consolidated
  - Renesas original 32bit RISC architecture
  - Renesas and ST adopt SuperH adopted for their device
  - SH-2, SH-2A, SH-3, SH-4, SH-4AL are not ABI compatible
  - SH-4A and ST-40 have backward compatibility to SH-4

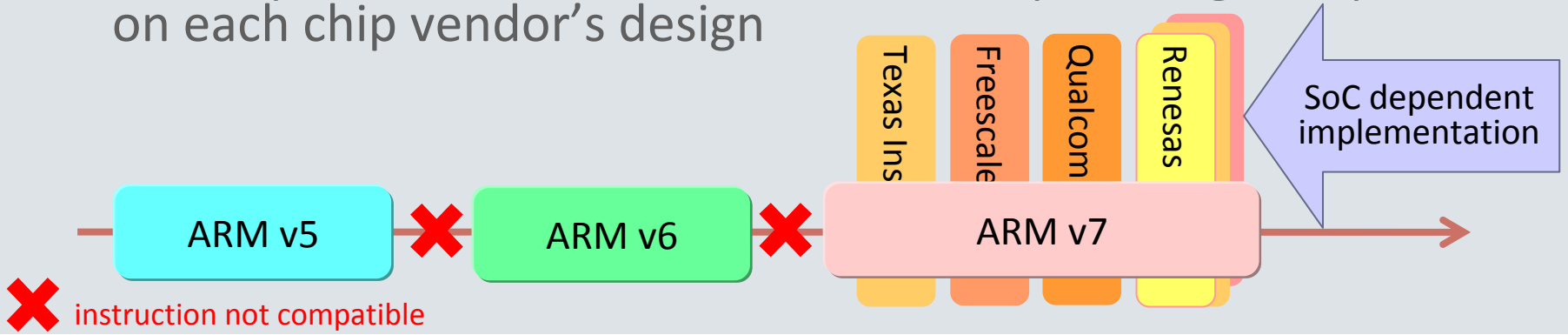


CPU flavor : ARM

- ARM : **Seems very hard to keep global consistency as there are too many players and variants**

ARM is adopted by many SoC vendors and its circumstances are much more complicated than SuperH.

- ARM company provide CPU architecture core to chip vendor
- ARM has different instruction set v5, v6 and now v7
- Later instruction set follows previous, but not fully compatible
- Interrupt controller connection to on-chip IPs might depends on each chip vendor's design



Agenda

- Is Linux getting fragmented due to embed device support ?
  - Device driver complexity ------ fat “defconfig” problem
  - Distribution fork ---------- Android kernel problem
  - CPU support confusion --------- vendor tree problem
- Lesson learned and our observation
  - SuperH kernel support (as architecture provider)
  - ARM kernel support (as ARM Linux newbie)
- Future direction of embedded Linux
  - How should we manage diversion of embedded Linux
  - Possible tactics

SuperH : kernel and toolchain

- SH-2, SH-2A, SH-3, SH-3DSP, SH-4AL(DSP) all have different core design and we need different kernel code for them. Linux kernel source can manage these variants support into one kernel source if we can define them properly. Nevertheless this requires huge effort and we are now concentrating on SH-4(A).
- In terms of GNU toolchain (complier, library and others), we also need to have a dedicated toolchain for each of them.
- These diversity makes hard to aggregate community effort for SuperH. Some community people still work on SH-2, Renesas is now concentrating on SH-4(a) development though.

We could concentrate on SH-4(A) for kernel / toolchain maintenance

SuperH : userland (distribution)

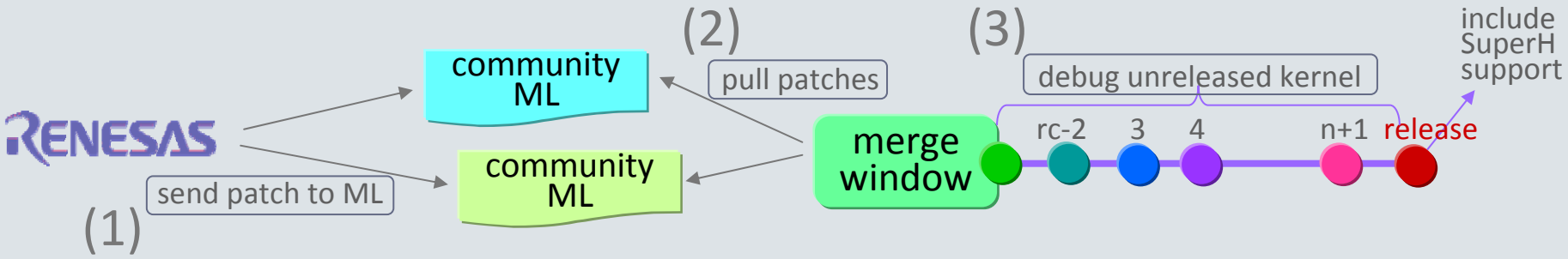
Renesas is engaged in Debian-ports project to provide fully maintained Linux pre-build applications that runs on SuperH.

- Formerly, Linux distribution like Redhat, Debian and SUSE have provided support for PC Linux (IA32) environment only.
- Now some major distribution (both commercial and community) start adding support for embedded CPU target as well.
- In these case, distributor expect to have common ABI (like IA32) for each CPU architecture. However Renesas do not have common ABI across all SuperH family, then we selected SH-4(A) as target architecture for SH Debian support..

Linux distribution expect one generic ABI for each CPU arch.

SuperH : Upstreaming (no private kernel tree)

- In terms of SH-4(A) , Renesas develops/maintains SuperH CPU support code and on-chip device driver code with upstream community. This means we submit all new code to appropriate community ML so that they can be pulled to at upstream merge timing, and we spend time for under-development version code test during its RC-1 to final release period.



- We do not need to manage own private kernel repository, as all required code are already part of upstream community code.

Still we needed some private code in BSP, why ?

- We aimed to migrate all SuperH code to the upstream, however we still need some private code to create our turnkey BSP.
- I know this is not a ideal state, but it worth highlight why still we needed to manage such private code.
  - Some driver code are still in review process in community. But at least we have submitted our code proposal to upstream.
  - Some code can not be merged due to its obsolete design.
  - Some industry initiatives not allow us to disclose userland library source code, like security authorization mechanism.
  - Chip vendor may deliver binary only firmware (=userland lib) for their IP block support, like paid video codec code.

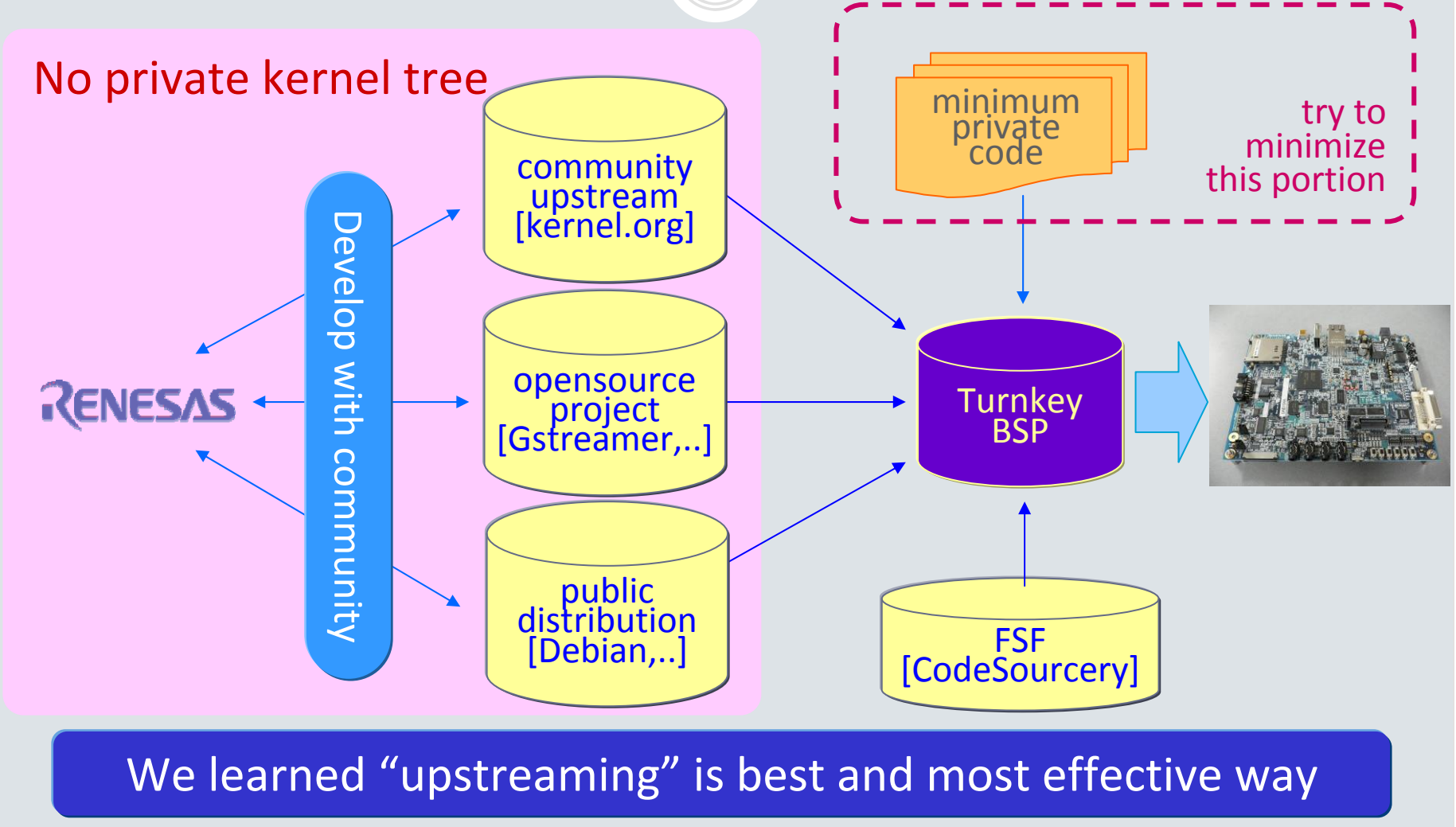
Lesson Learned (1) : SuperH is relatively simple world

- SuperH development is quite straightforward, because

  - SuperH is our in-house CPU core, we can easily access to device design team to confirm any unknown hardware issue.
  - SuperH ABI designed inside Renesas, and we can modify it if any enhancement needed like additional elf-fdpic support.
  - Now we have upstream SuperH architecture maintainer, who is Paul Mundt, inside Renesas. He is acting as direct gateway between chip vendor and upstream kernel community.

We could manage SuperH not to have excessive diversity

Lesson Learned (2) : How to eliminate off-tree code



We learned “upstreaming” is best and most effective way

Agenda

- Is Linux getting fragmented due to embed device support ?
  - Device driver complexity ------ fat “defconfig” problem
  - Distribution fork ---------- Android kernel problem
  - CPU support confusion --------- vendor tree problem
- Lesson learned and our observation
  - SuperH kernel support (as architecture provider)
  - ARM kernel support (as ARM Linux newbie)
- Future direction of embedded Linux
  - How should we manage diversion of embedded Linux
  - Possible tactics

We tried “no local repository” strategy for ARM, but…

- We tried to apply exact same way for our ARM integrated SoC Linux support, as we believe that is the best practice.
- However,… we found

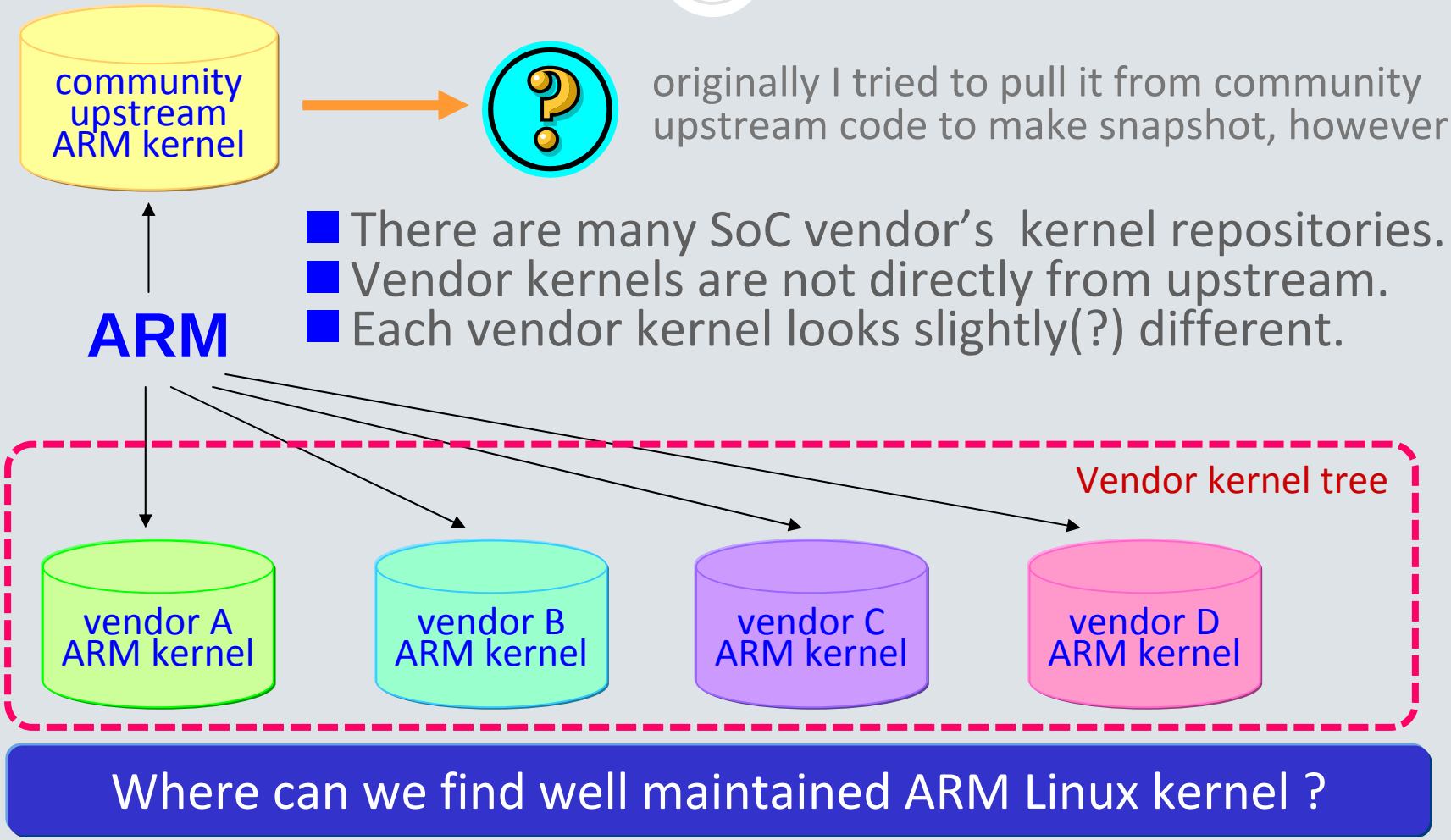
Vendor tree issue

- Latest ARM support code is not fully integrated to upstream ARM kernel so far.
- There are various vendor kernel tree and they include some important patch to support ARM core.

New findings for us

- Our ARM support patch may break existing kernel code due to ARM instruction set and/or hardware incompatibility.

Vendor tree diversity



Why so many vendor trees ?

- Local driver code that can not be integrated to upstream. This is basically code quality issue and this can be solved.
- Want to stick on some specific old kernel version because they do not need to catch up latest kernel migration.
- Each chip vendor need to apply device workaround patch before its merge to upstream. And workaround patches written by each SoC vendor is not exactly same.
- All patches can not be integrated due to ABI incompatibility. As seen on SH-2/3/4, there are some APB/ABI/hardware incompatibility on various generation ARM processor.

Case study (1) : Does Renesas break current ARM world ?

- We start submitting SH-MobileAP series kernel and driver patch to ARM kernel community upstream.
- We tried to add DMA engine support for ARM v7 based Cortex A8 processor to support that require multiple memory mappings with deferent memory type.
- ARM maintainer, Russell King pointed out our DMA engine code will break existing ARM code, as it is an architectural restriction. (See next page for actual message)
- Then we proposed new DMA design that can work on both existing device and SH-Mobile and Russell accepted, **however** ….

I think this example implies the root cause of vender tree.

Case study (2) : We hit ARM architectural restriction

http://marc.info/?l=linuxsh&m=128130485208262&w=2

From: Russell King - ARM Linux <linux@arm.linux.org.uk>
To: Arnd Hannemann arnd@arndnet.de
Cc: linux-arm-kernel@lists.infradead.org, linux-sh@vger.kernel.org
Subject: Re: [regression] in linux-next: sh_mobile_ceu_camera broken by "ARM: Prohibit ioremap() on kernel managed RAM"
Date: Sun, 08 Aug 2010 23:00:35 +0100
Sender: linux-sh-owner@vger.kernel.org
User-Agent: Mutt/1.5.19 (2009-01-05)

Me neither! However, reverting the commit isn't the answer.

ARM shmobile platforms are ARM architecture V6 or V7, both of which have the architectural restriction that multiple mappings of the same physical address space must have the same memory type and cache attributes. We know that some ARMv7 systems do bad things when multiple different mappings exist - and as they're using ARM's own Cortex CPU designs, and I doubt shmobile is any different in that...

So, basically going forward with the advent of VIPT caches on ARM, any kernel interface which allows system RAM to be remapped with different attributes from the lowmem mapping is bad news - that means (eg) using vmap() with a non-PGPROT_KERNEL pgprot has become illegal. Certainly using ioremap() on mapped system RAM on VIPT has become illegal, and that's what has now been prevented.

It's actually a restriction which x86 gained some time ago, which I stupidly continued to permit on ARM. Now that our hardware has gained the same restriction, we're now going to be into the same learning curve...

Case study (3) : unexpected propagation of discussion

http://marc.info/?l=linux-sh&m=128297954820636&w=2

From: FUJITA Tomonori <fujita.tomonori@lab.ntt.co.jp>
To: mitov@issp.bas.bg
Cc: fujita.tomonori@lab.ntt.co.jp, u.kleine-koenig@pengutronix.de, g.liakhovetski@gmx.de, linux-kernel@vger.kernel.org, linux-media@vger.kernel.org, akpm@linux-foundation.org, linux-arm-kernel@lists.infradead.org, linux-sh@vger.kernel.org, philippe.retornaz@epfl.ch, gregkh@suse.de, jkrzyszt@tis.icnet.pl
Subject: Re: [RFC][PATCH] add dma_reserve_coherent_memory()/dma_free_reserved_memory() API
Date: Sat, 28 Aug 2010 16:10:28 +0900
Sender: linux-sh-owner@vger.kernel.org

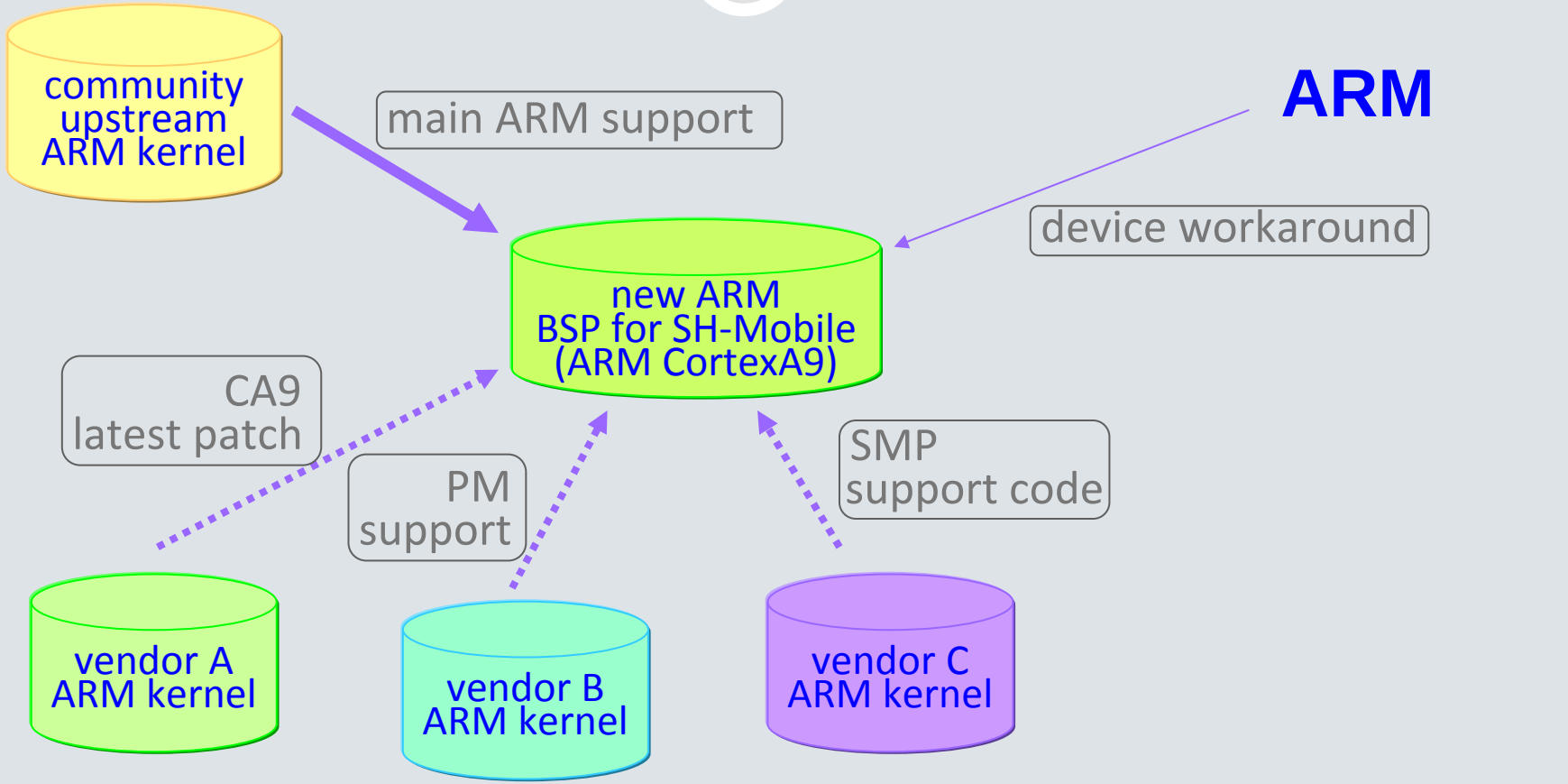
I think that I already NACK'ed the patch.

(snip)

IMHO, reverting the commit 309caa9cc6ff39d261264ec4ff10e29489afc8f8 temporary (or temporary disabling it for systems that had worked) is the most reasonable approach. I don't think that breaking systems that had worked is a good idea even if the patch does the right thing. I believe that we need to fix the broken solution (videobuf-dma-contig.c) before the commit.

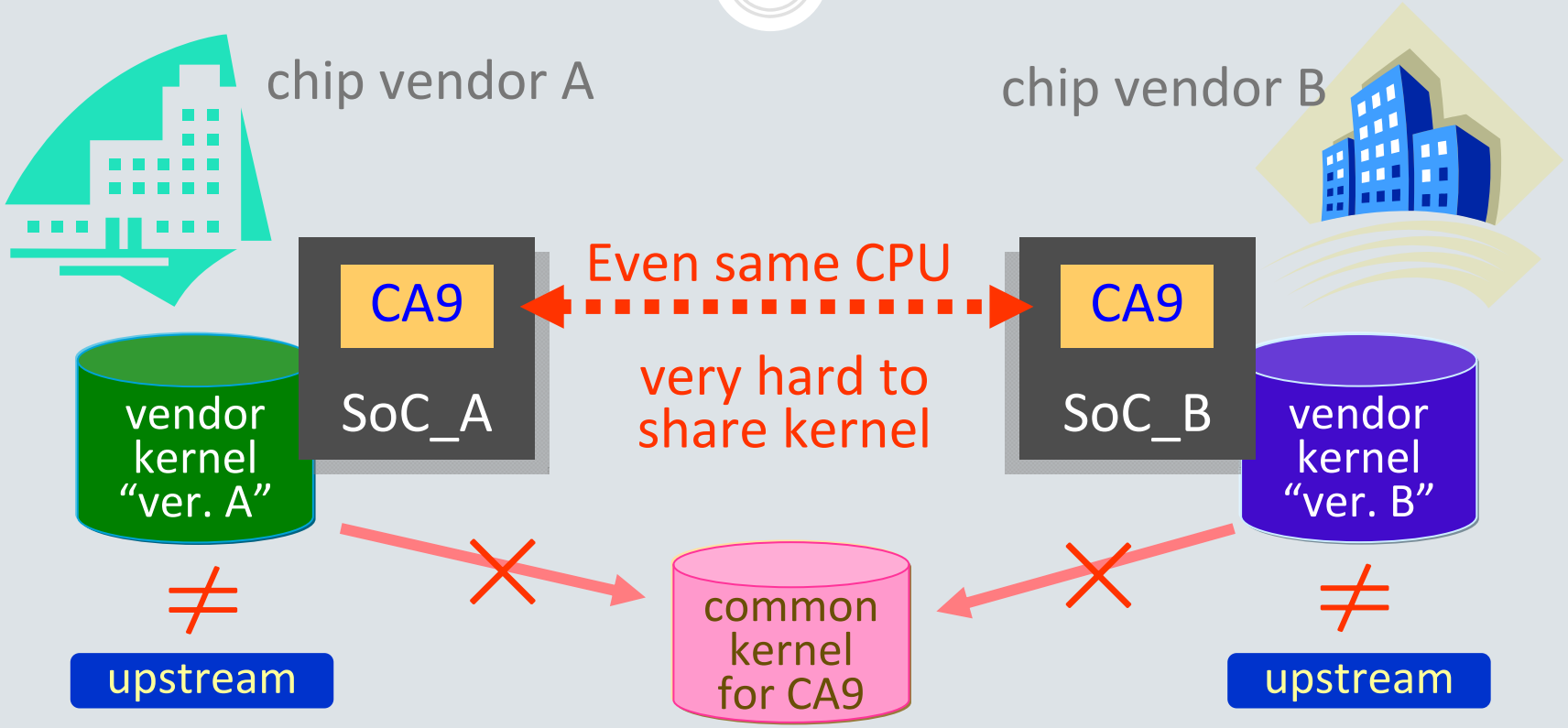
Our “technically right ” patch breaks existing Linux system,
I wonder this can be a enough motivation to have local tree.

Lesson Learned (3) : SoC vendor's problem



Need to cherry pick from various ARM kernel repository

Lesson Learned (4) : ARM SoC adopter's problem



User can not utilize ARM experience across other vendor SoC even on same CPU core if it comes from different chip vendor

Lesson Learned (5) : This is not ARM specific issue

Initially I thought this is ARM specific problem and never seen on other architecture. However I was reminded that this type of confusion happens also on other embedded CPU including Intel based SoC at CELF Japan Jamboree.

- CPU architecture provider like ARM and MIPS can not manage each SoC dependent implementation part code. Also peripheral IP selection is of course fully up to each SoC vendor.

- As embedded does not have BIOS (hw. abstraction layer) , people need to have each dedicated defconfig file. And this is exactly same on Intel based embedded platform.

Embedded CPU/SoC/Platform support is not same as PC based, and without special care it tend to bloat, fragment and diffused.

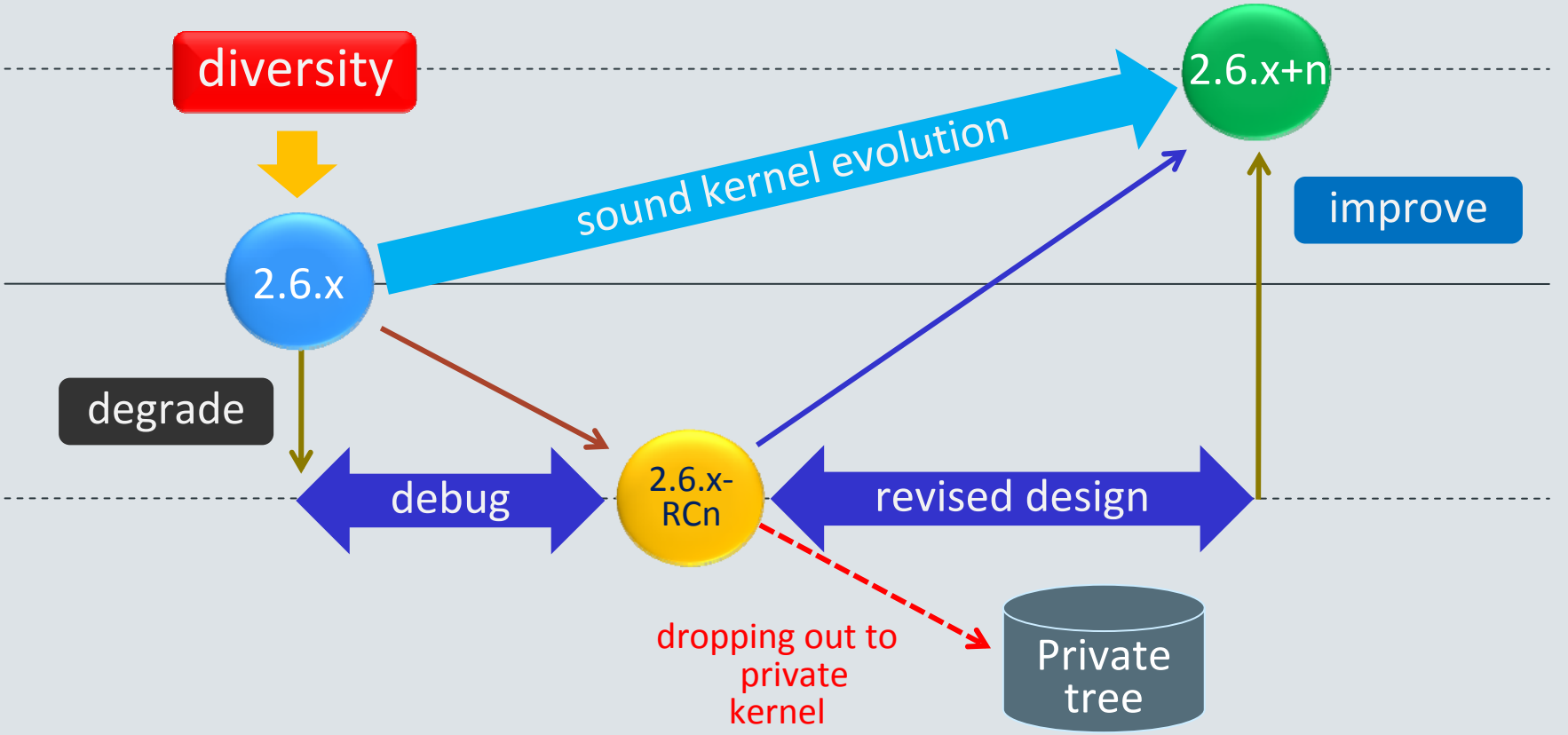
Agenda

- Is Linux getting fragmented due to embed device support ?
  - Device driver complexity ------ fat “defconfig” problem
  - Distribution fork ---------- Android kernel problem
  - CPU support confusion --------- vendor tree problem
- Lesson learned and our observation
  - SuperH kernel support (as architecture provider)
  - ARM kernel support (as ARM Linux newbie)
- Future direction of embedded Linux
  - How should we manage diversion of embedded Linux
  - Possible tactics

Direction of future embedded Linux (idea)

- Linux kernel enhancement to add various embedded device is great thing, and we heavily depend on it now.
- However uncontrolled enhancement might cause loss of whole eco-system harmonization as already seen on ARM SoC support.
- I know current Linux patch review/accepting works quite well, but I think we should start thinking about how to manage healthy growth of embedded SoC support in Linux right now. As this may require seamless alliance across competing SoC vendor using same CPU core, organization like LF or CELF need to initiate this kind of consolidation process.
- Also each SoC vendor should try to eliminate private tree diversion not to cause isolation from mainstream Linux growth.

Sound dialectic process will make evolution



Diversity (and right handling) is a mother of sound Linux evolution

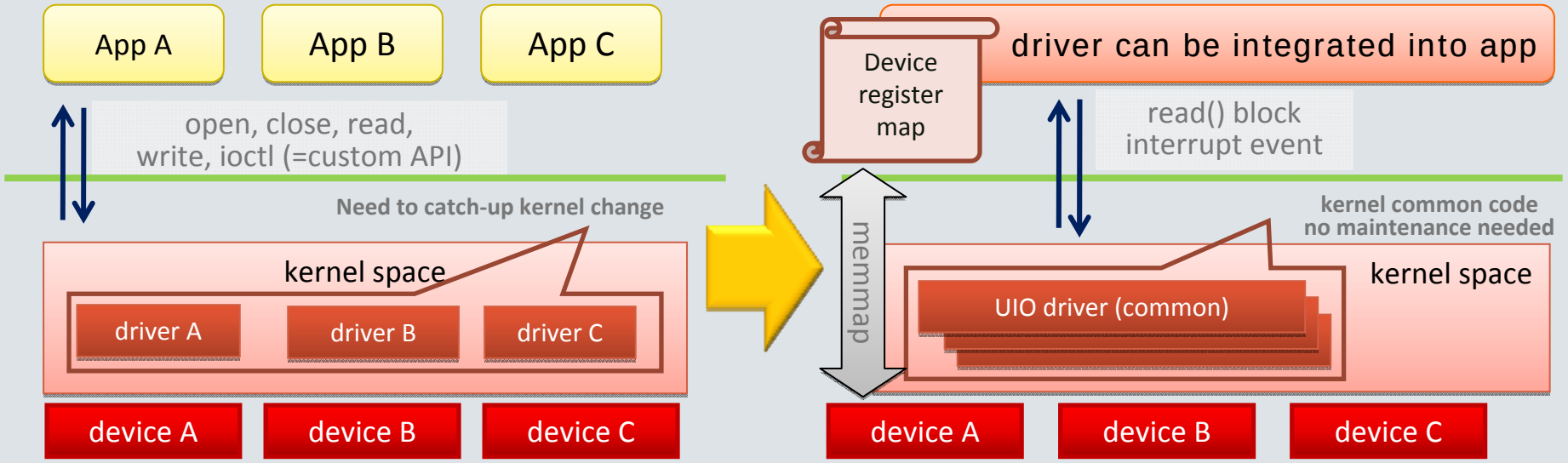
Possible tactics (1) : code cleanup mechanism

- To avoid unregulated bloat of embedded device support, we should apply some sort of automated cleanup mechanism, as not all platform / device are maintained at every kernel migration.
  - How to detect dead platform / device.
  - How to drop support for these target.
- Also we should think about more code share across platform.
  - ARM device tree is one good example.
  - IP compatibility is also expected not to cause similar variant.

Possible tactics (2) : UIO adoption

- SoC unique device driver can use UIO to eliminate bloat of kernel driver code. With UIO all driver can share common UIO driver and actual device handle can be implemented in each userland.

- This is a kind of trade off of openness and cleanup, but at least non-general device can be supported with UIO method without bother common kernel space code.

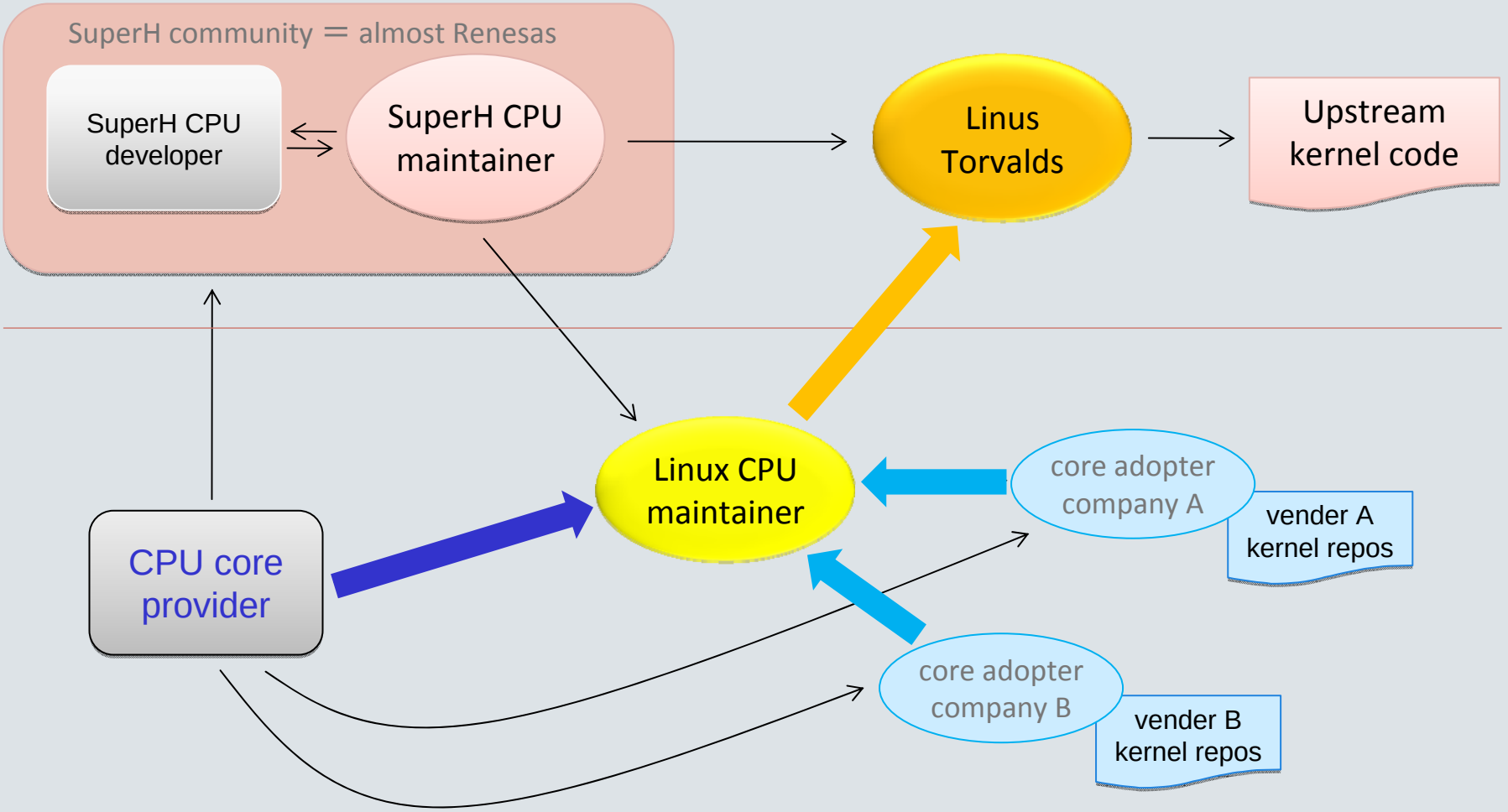


Possible tactics (3) : Help maintainer's work

- CPU architecture maintainer owns heavy burden
  - Apply any new kernel capability to his architecture
  - Apply latest CPU core enhancement to upstream
  - Apply any CPU core workaround code to upstream
  - Coordinate various instruction set
  - Merge each vendors CPU implementations, like clock and interrupt controller driver.
- These work can not be completed if he can not get enough information share with CPU core provider company.

CPU and SoC vendor should work closely with maintainer

Possible tactics (3) : Help maintainer's work



Conclusion

- Upstreaming is the most effective way to utilize opensource. We have applied this on our SuperH with some device support coverage consolidation.
- When we start working on ARM core SoC, we noticed that ARM world is bit fragmented than SuperH as it has many vendor trees and bloated defconfig / local device that Linus claims.
- Now we reminded this ARM Linux confusion can be common embedded SoC issue that might happen on any SoC type device.
- I think we should start some coordination work to avoid further unmanaged diversion and bloat of embedded device support in Linux not to cause detachment from mainstream evolution.