devmatch – Matching Devices to Modules

M. Warner Losh

Netflix, Inc.

BSDCan 2018

NETFLIX

http://people.freebsd.org/~imp/talks/bsdcan2018/bsdcan2018.pdf

BSDCan 2018
The Technical BSD Conference
High value. Low cost. Something for everyone.

Unix Questions

Q: How big was the first PDP-11 Unix kernel we have?

Unix Questions

Q: How big was the first PDP-11 Unix kernel we have?
A: About 9k (8960 bytes) (snapshot between V1 and V2)

Unix Questions

Q: How big was the first PDP-11 Unix kernel we have?
A: About 9k (8960 bytes) (snapshot between V1 and V2)

Q: How big is FreeBSD/i386 11.1 GENERIC kernel?

Unix Questions

Q: How big was the first PDP-11 Unix kernel we have?
A: About 9k (8960 bytes) (snapshot between V1 and V2)

Q: How big is FreeBSD/i386 11.1 GENERIC kernel?
A: Almost 24MB! (23757224 bytes)

Unix Questions

Q: How big was the first PDP-11 Unix kernel we have?
A: About 9k (8960 bytes) (snapshot between V1 and V2)

Q: How big is FreeBSD/i386 11.1 GENERIC kernel?
A: Almost 24MB! (23757224 bytes)

Q: How fast has the FreeBSD/i386 kernel grown?

Unix Questions

Q: How big was the first PDP-11 Unix kernel we have?
A: About 9k (8960 bytes) (snapshot between V1 and V2)

Q: How big is FreeBSD/i386 11.1 GENERIC kernel?
A: Almost 24MB! (23757224 bytes)

Q: How fast has the FreeBSD/i386 kernel grown?
A: About 20% per year – About half the Moore's Law rate

Outline

Outline

Motivation for Work

- GENERIC size
  - Grown from 0.5M to 26MB in 25 years
- Compile time growth
- Load time growth (especially netboot)
- Reduce redundancy
  - Most monolithic drivers also built as modules
  - Build system has not evolved as promised
- Eases integration of 3rd party drivers
- Why not – It's Cool

Outline

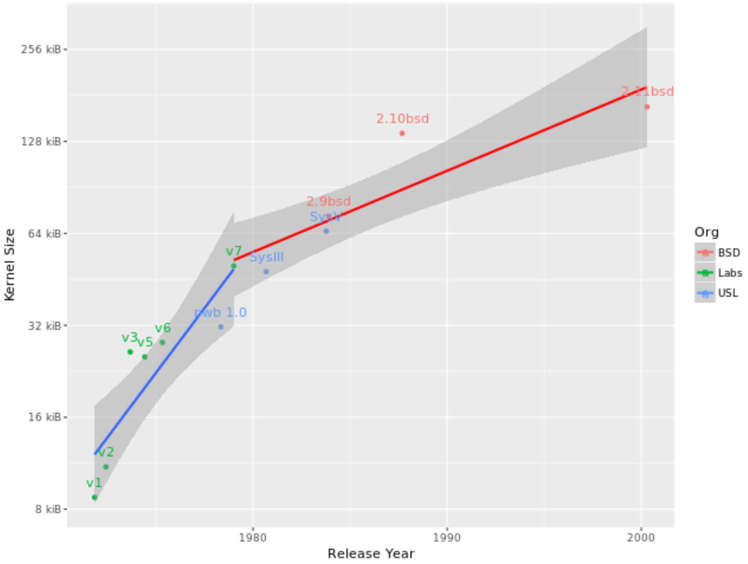
Kernels Grow

- Unix Kernel size has grown
- Growth rate has been exponential
- Proliferation of drivers
  - V7 Unix had 22 drivers
  - FreeBSD 12 has $\sim$ 1700 drivers ($\sim$ 380 FDT, $\sim$ 330 PCI)
- Proliferation of technology stacks
  - Research Unix barely had networking
  - FreeBSD has TCP/IP, sockets, SATA, SCSI, NVMe, IPv6, DMA, IPSEC, firewalls, iSCSI, ATM, PCIe, Crypto, etc
- Compilers have gotten better (eg, more inlining makes faster code)
- Most of the kernel functions can come from modules

Research Unix

- Original PDP-11 Unix from Bell Labs
- Not all early versions are still extant
- Limit for V1, V2 and V3 was 16k due to C compiler constraints
- Rewrite from assembler to C happened in v3-v4
- Size constrained by extreme memory prices
- No GENERIC-like kernel, config was compiled in.
- BSD 2.x and System III/V included
- Spans 30 years: Growth rate 15%/year
- V2–V7 growth rate 28%/year
- Data from TUHS (`http://www.tuhs.org`)

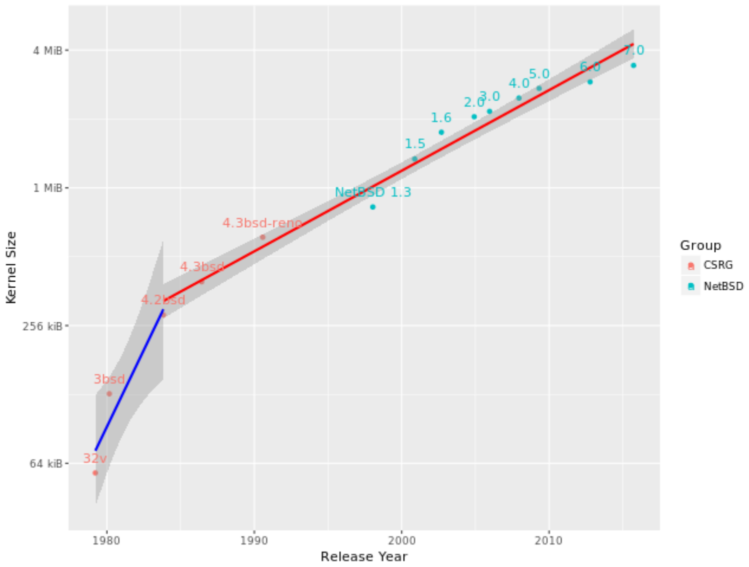
PDP-11 Kernel Size



BSD Unix

- VAX Releases from 32V onward
- Very fast growth to accommodate paging and sockets / networking
- Exponential growth from 4.2BSD onward
- No 4.4BSD VAX image
- NetBSD/vax used post CSRG disbanding
- Spans almost 40 years: Growth rate 10%/year

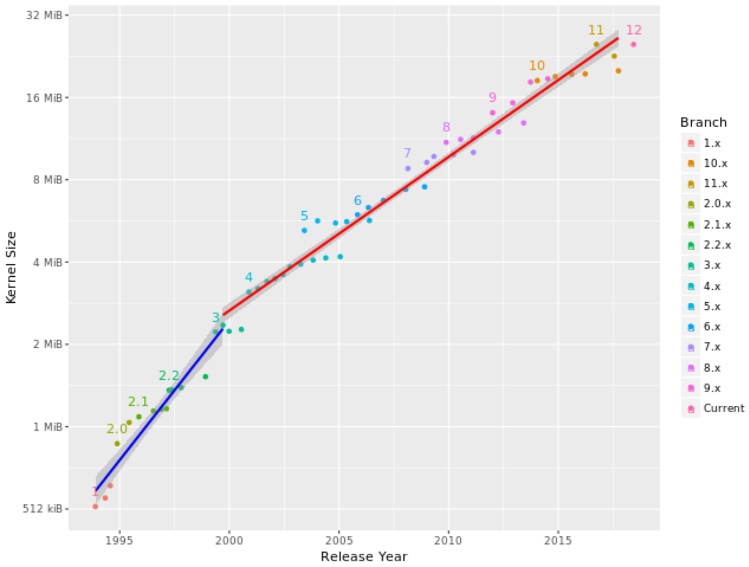
VAX Kernel Size



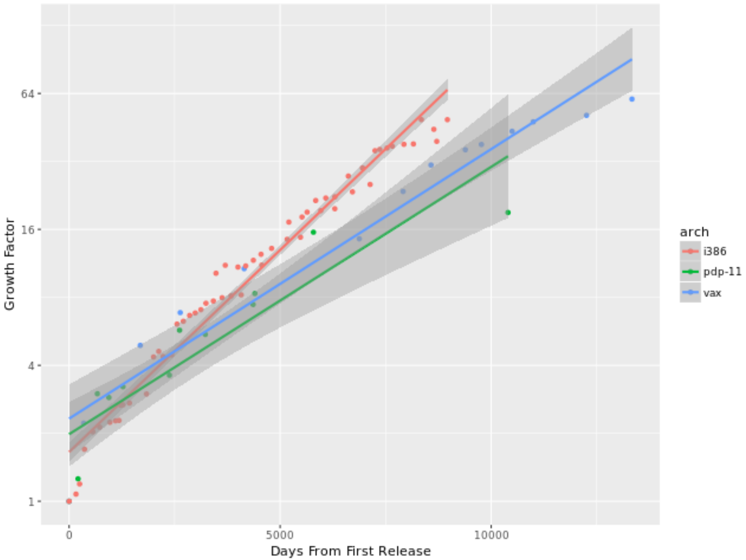
FreeBSD/i386

- Size of GENERIC kernel from release media
- GENERICBH used before 2.0
  - No GENERIC, kernel too big for 640k
  - Limited driver support
- Size not normalized to a specific compiler
- Size dipped between 11.0 and 11.1 due to clang bump
- Spans almost 30 years: growth rate 20%
- Grew faster, proportionally, betweeen 1.0 and 3.2 (31%/year)

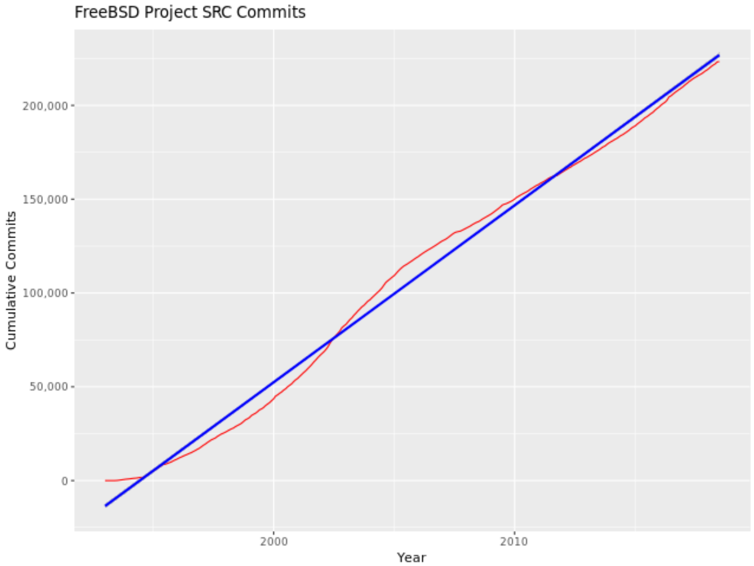
FreeBSD/i386 Kernel Size



Normalized Kernel Growth



FreeBSD Commits To Date



Kernel Size Redux

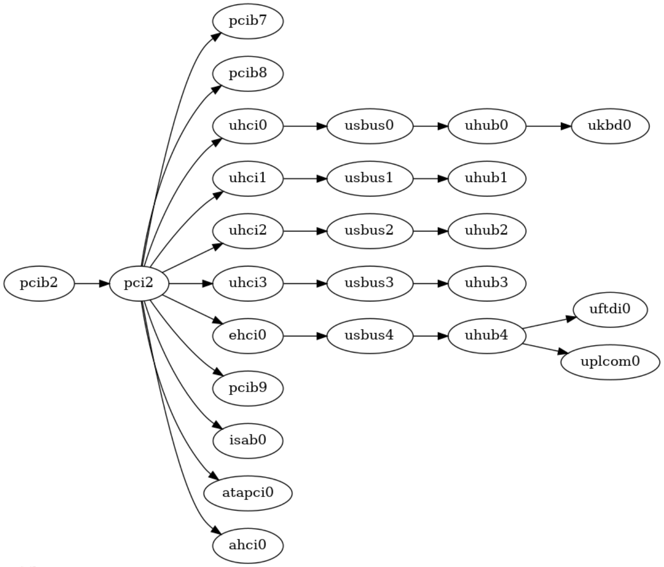
Kernel Series	Years	Rate	Doubling Time
Research (V1–V7) Unix	7	28%	2.5 Years
AT&T PDP-11	28	10%	7 Years
Early BSD VAX	29	11%	6.8 Years
BSD VAX	4.5	43%	19 Months
FreeBSD/i386 1.0 – 3.0	5.5	31%	27 Months
FreeBSD/i386 3.0 – 11.0	18	15%	4.6 years
FreeBSD/i386 1.0 – 11.0	24	20%	3.5 years
Moore's Law	50?	35%	24 Months

Outline

Newbus details

- Tree hierarchy
  - A bus is just a driver with children
  - Buses supply pnpinfo
- Self enumerating bus
  - All have per–device matching drivers to devices
  - Generically called pnpinfo
- Hinted bus
- Probe routines
  - Examine pnpinfo to see if driver matches
  - Some buses centralize probe, others are ad-hoc

Simplified Newbus Device Tree



Typical Probe Routine (good)

```
static const struct ral_pci_ident ral_ids[] = {
        { 0x1432, 0x7708, "Edimax␣RT2860" },
... };

static int ral_pci_probe(device_t dev)
{
   const struct ral_pci_ident *ident;
   for (ident = ral_ids; ident->name != NULL; ident++) {
      if (pci_get_vendor(dev) == ident->vendor &&
          pci_get_device(dev) == ident->device) {
         device_set_desc(dev, ident->name);
         return (BUS_PROBE_DEFAULT);
      }
   }
   return ENXIO;
}
```

Typical Probe Routine (bad)

```
static int nvme_probe (device_t device)
{
...
    while (ep->devid) {
        if (nvme_match(devid, subdevice, ep)) {
            device_set_desc(device, ep->desc);
            return (BUS_PROBE_DEFAULT);
        }
        ++ep;
    }
    if (pci_get_class(device)    == PCIC_STORAGE &&
        pci_get_subclass(device) == PCIS_STORAGE_NVM &&
        pci_get_progif(device)   == NVM_NVMHCI_1_0) {
            device_set_desc(device, "Generic␣NVMe␣Device");
            return (BUS_PROBE_GENERIC);
    }
    return (ENXIO);
}
```

Crazy Probe Routine

```
static int
tulip_pci_probe(device_t dev)
{
    const char *name = NULL;
    if (pci_get_vendor(dev) != DEC_VENDORID)
        return ENXIO;
    if (pci_get_subvendor(dev) == 0x1376)
        return ENXIO;
    switch (pci_get_device(dev)) {
    case CHIPID_21040: name = "21040␣Ethernet"; break;
    case CHIPID_21041: name = "21041␣Ethernet"; break;
    case CHIPID_21140: name = "21140A␣Fast␣Ethernet"; break;
    case CHIPID_21142: name = "21143␣Fast␣Ethernet"; break;
    }
    if (name) {
        device_set_desc(dev, name);
        return BUS_PROBE_LOW_PRIORITY;
    }
    return ENXIO;
}
```

Module details

- Metadata placed in the code to mark modules
- What version, what depends, how to connect to newbus
- Metadata post-processed by kldxref(8)
- SYSINITs that force a probe on kldload(8) and kldunload(8)

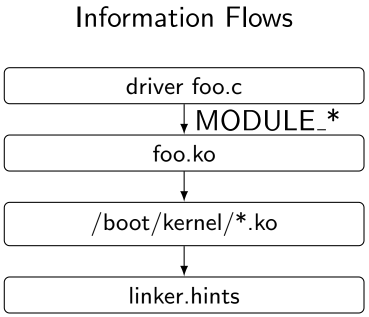
Typical Module Marking

```
MODULE_DEPEND(ral, pci, 1, 1, 1);
MODULE_DEPEND(ral, firmware, 1, 1, 1);
MODULE_DEPEND(ral, wlan, 1, 1, 1);
MODULE_DEPEND(ral, wlan_amrr, 1, 1, 1);
DRIVER_MODULE(ral, pci, ral_pci_driver, ral_devclass,
    NULL, NULL);
```

kldxref(8)

- Parses module metadata out of .ko files
- Creates /boot/kernel/linker.hints
  - Contains module name to file name mapping
  - Contains module dependency information
  - Contains module version information
  - Contains newbus attachment information
- Usually run at ‘make installkernel’ time.
  - Now run at boot since kldxref(8) is only native

kldxref data flow



buildkernel

installkernel

kldxref

rc.d(8)

- FreeBSD's init scripting system
- Scripts run in dependency order at boot to start services
- Flexible and extensible
- Post–boot modules currently installed

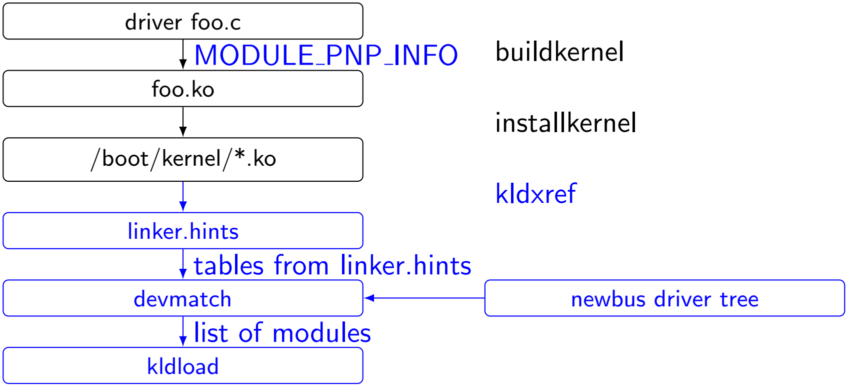
devd(8)

- Reacts to generic events from the kernel
- Runs scripts when devices found, GEOM devices appear, etc
- One event is ‘driver NOMATCH’ when no driver claims a device

Outline

Data Flow for Changes

Information Flows



Design Overview

- Mark driver's PNP information
- Extend kldxref(8) to understand new markings
- Write program to parse linker.hints and compare to system (devmatch)
- Write new rc.d script to glue it together
- Write new devd.conf rules
- Fix module penalty
- Extensions to newbus
- Tune MINIMAL and transition to reduced GENERIC

PNP Info Decoration

- Assumes we have a table
- Describes the table
  - Size of each entry
  - Number of entries
  - Format of each entry
- Leverages off the module marking system
- Designed for smooth transition
- Buses with centralized probe have wrapper macro

Table Description Details

- ASCII string
- One or more instances of ‘TYPE:FIELD;’
- TYPE on next slide and MODULE_PNP_INFO
- FIELD is either the name of bus specific pnpinfo item or ‘#’
- Examples:
  - U16:vendor;U16:device;
  - W32:vendor/device;
  - D:#;V32:manufacturer;V32:product;Z:cisvendor;Z:cisproduct;

Type Language

Type	Description
U8	uint8_t element
V8	uint8_t but 0xff matches all
G16	uint16_t greater than or equal
L16	uint16_t less than or equal
M16	uint16_t mask of fields that follow to use
U16	uint16_t
V16	uint16_t but 0xffff matches all
U32	uint32_t
V32	uint32_t but 0xffffffff matches all
W32	two uint16_t field/field as one word host order
Z	ASCII string terminated by NUL
D	Description NUL terminated
P	Pointer sized thing that's ignored

Typical Change (Centralized Probe)

```
static const STRUCT_USB_HOST_ID uark_devs[] = {
  {USB_VPI(USB_VENDOR_ARKMICRO,
    USB_PRODUCT_ARKMICRO_ARK3116, 0)},
};

DRIVER_MODULE(uark, uhub, uark_driver, uark_devclass,
  NULL, 0);
MODULE_DEPEND(uark, ucom, 1, 1, 1);
MODULE_DEPEND(uark, usb, 1, 1, 1);
MODULE_VERSION(uark, 1);
+USB_PNP_HOST_INFO(uark_devs);
```

Typical Change (Ad Hoc Probe)

```
static struct _pcsid
{
        uint32_t        type;
        const char      *desc;
} pci_ids[] =
{
        { 0x140111f6, "Compex␣RL2000" },
...
        { 0x00000000, NULL }
};
...
+MODULE_PNP_INFO("W32:vendor/device;D:#", pci, ed, pci_ids,
    nitems(pci_ids) - 1);
```

Typical Change (Crazy Probe)

STANDING OUT IN A CROWD
ISN'T NECESSARILY A GOOD THING
imgflip.com
©2005 Jessica and John Williams

Typical Change (Crazy Probe)

```
pci_ids[] = {
 { (CHIPID_21040 << 16) | DEC_VENDORID, "21040␣Ethernet" },
 { (CHIPID_21041 << 16) | DEC_VENDORID, "21041␣Ethernet" },
 { (CHIPID_21140 << 16) | DEC_VENDORID, "21140A␣Ethernet" }
 { (CHIPID_21142 << 16) | DEC_VENDORID, "21143␣Ethernet" },
 { 0x00000000, NULL }
};
static int tuplip_pci_probe(device_t dev) {
 uint32_t       type = pci_get_devid(dev);
 struct _pcsid  *ep =pci_ids;
 while (ep->type && ep->type != type)
   ++ep;
 if (ep->desc == NULL)
   return (ENXIO);
 device_set_desc(dev, ep->desc);
 return (BUS_PROBE_DEFAULT);
}
...
+MODULE_PNP_INFO("W32:vendor/device;D:#", pci, de, pci_ids,
    nitems(pci_ids) - 1);
```

newbus Freeze and Thaw

- Enhance newbus to understand deferring of probing
- Need to wait for all drivers to load
- For each module loaded, add to deferred probe list
- When thawed add all drivers in the list to the system
- Once all new drivers are added, trigger driver_added callbacks

kldxref(8) Changes

- Add code to parse new MODULE_PNP_INFO nodes in .ko's
- Convert the tables to a simplified form
- Write out the new tables extracted from the binary to linker.hints

linker.hints Type Info

Type	Description
I	int
J	int (-1 means ignore)
G	int (greater than or equal)
L	int (less than or equal)
M	int (mask)
D	Description
Z	Ascii string
T	value true for all elements in table

devmatch(8) Program

- Parses linker.hints
- Gets driver tree from kernel
- Walks the tree looking for different issues
  - Unattached devices that may match one or more modules
  - Attached drivers that don't match a module
  - All device
  - Dump linker.hints file
- Defaults to /boot/kernel/linker.hints, but can look at any linker.hints file (.ko's need not be present)
- Can run with just the devmatch NOMATCH string

devmatch rc.d script

- Simple script running devmatch
- Sorts the output and discards duplicates
- Freezes newbus
- loads all the .kos
- thaws newbus

devmatch devd script

- Simple NOMATCH script that passes the NOMATCH string to devmatch

```
# Generic NOMATCH event
nomatch 100 {
        action "/etc/rc.d/devmatch start '?$_'";
};
```

MINIMAL kernel

- Removes all drivers that aren't root or console devices
- Root devices could be found by /boot/loader, but aren't today
- Root devices may have other dependencies (eg root is on MPT card, but also needs CAM)
- Console devices can't be loaded modules because cninit() runs before module list from loader processed

Boot Loader Futures

- linker.hints is read in by /boot/loader today
- We skip the pnp info tables
- Future versions could load all storage devices as possible sources of root.

Google Summer of Code

- Lakhan Kamireddy
- `https://wiki.freebsd.org/SummerOfCode2018Projects/ConvertPCIdriverAttachmentsToTables`
- Good progress. Commits in tree. 30 more changes after talk.
- About 380 PCI drivers in tree
- About 300 are entirely table driven, 50 more are close, 30 others are troublesome (eg if_de) in some way.

Outline

Long PNP Info

- Old interface truncated pnpinfo at 128 characters
- Parts of lost strings needed for many USB devices
- Fix is to export strings in string table rather than fixed sized array
- libdevinfo ABI didn't need to change

Multiple Instances of Devices

- First iterations lacked sort / unique step
- Modules loaded many times
- Partially backed out to fix USB issues with ums/uhid

Multiple Matching Drivers

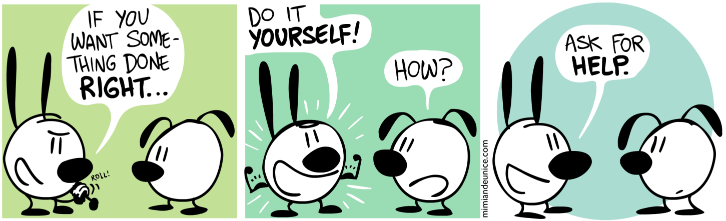
- Sometimes multiple drivers match
- Without Freeze/Thaw, first one will win
- Freeze/Thaw pending testing

Lots of Legacy Drivers

- We have lots of legacy drivers in the tree
- Many of them are not table driven
- Many PCI drivers don't use centralized routine
- GSoC student converting PCI drivers
- About 40 / 380 drivers done

384 FDT Drivers

- Still have lots of FDT drivers that need conversion
- NO GSoC student converting FDT drivers
- You can help! Ask me how.



Source: `http://mimiandeunice.com/2010/08/02/d-i-y/`

Module Penalty

- Modules that load have small performance penalty
- atomics not inlined
- locking not inlined
- On amd64, code is pic, which runs slower
- People that have measured say there's little difference despite these things

ATA PCI Driver

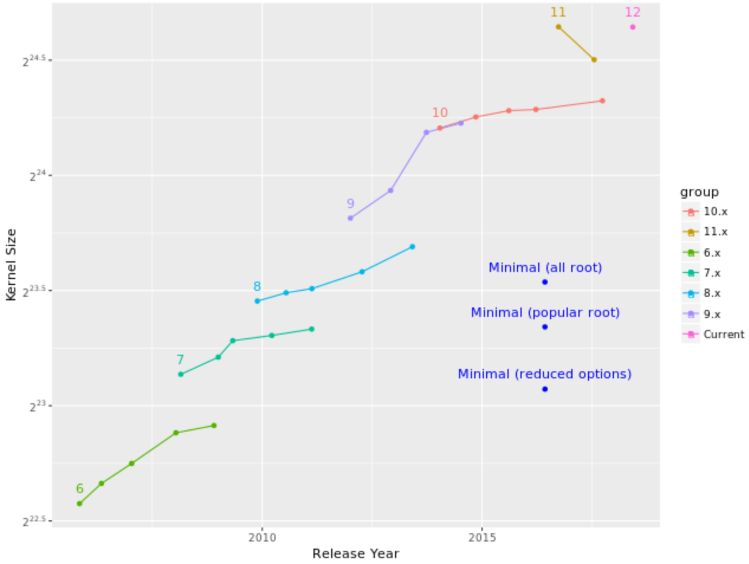
- Matches on class, subclass and any revid
- PCI publishes class, subclass and revid as one number
- Need a mask to specify which part of the PCI 'class' to match
- ATA PCI devices use revid as a bitmask, so all combos valid
- Likely need to create a new type to mask a field (existing mask type is mask of which fields are valid)
- Sadly, it's not the only weird edge case

Open Issues

- Newbus freeze/thaw
- Lingering uhid/ums issues
- ata pci mask issue
- 64-bit W64 may be needed, other types too
- Multiple linker.hints files
- Lots of drivers need a small amount of love
- MINIMAL tuning and testing (replace GENERIC?)
- Module Penalty?
- Cross build support for kldxref(8)
- Multiple MODULE_PNP_INFO entries

Outline

FreeBSD kernel Size Redux



FreeBSD kernel Size Redux

Kernel	Size	% Smaller	Wayback
Minimal all roots head	12170464	54%	Mid 8.x
Minimal popular roots head	10633696	59%	Late 7.x
Minimal reduced options	8818214	66%	Early 7.x
GENERIC 11.1	23757224		
GENERIC head	26211080		

Questions

Questions?
Comments?

Warner Losh

wlosh@netflix.com
imp@FreeBSD.org
@bsdimp

http://people.freebsd.org/~imp/talks/bsdcan2018/bsdcan2018.pdf