



A High-Performance Driver Ecosystem for NVM Express

Andy Rudoff
Datacenter Software
Intel



Ecosystem Goals

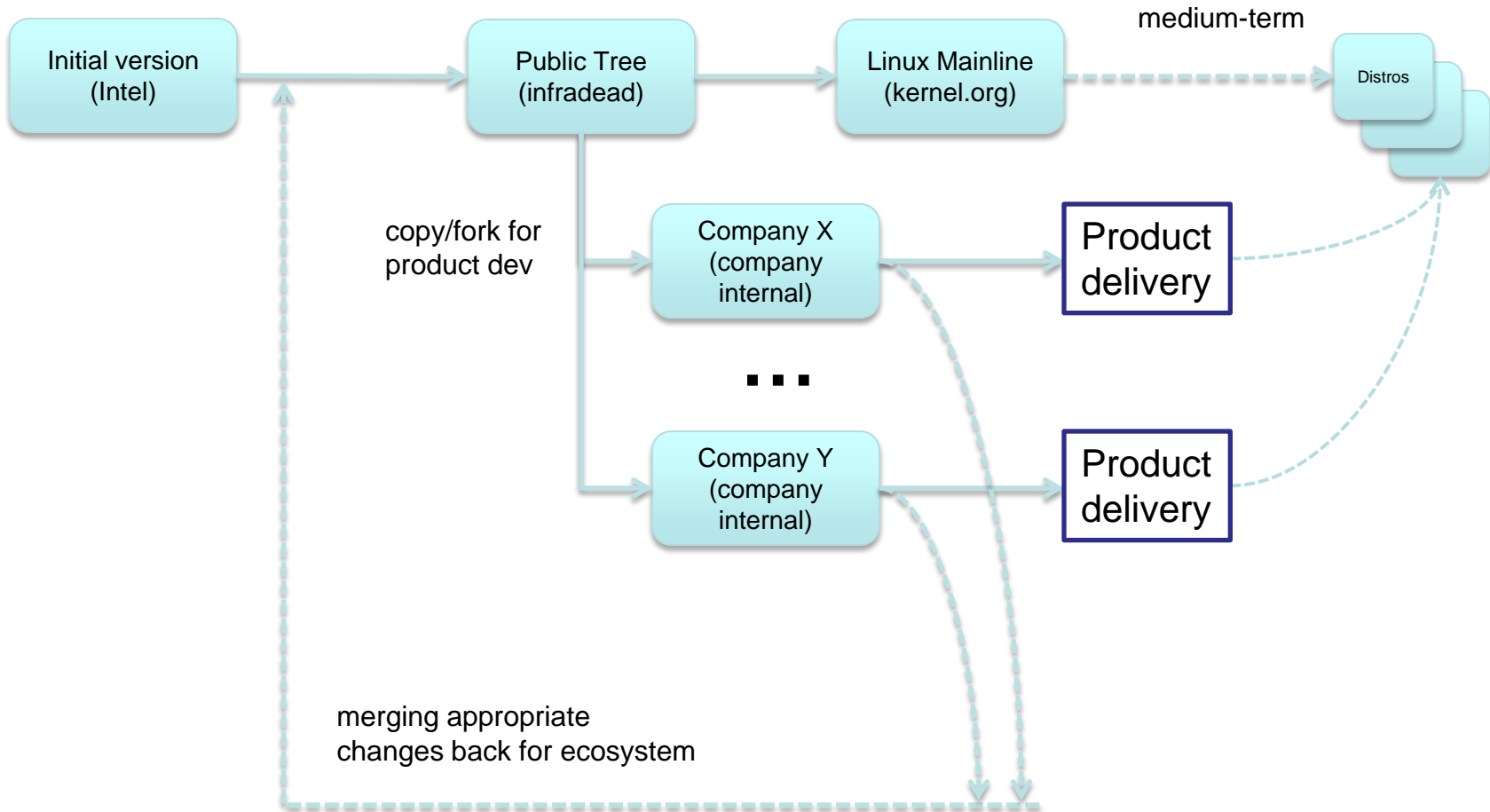
- Long-term
 - Each OS comes with standard NVMe driver

- Short-term
 - Allow NVMe vendors to provide the drivers they need with their products without major driver development

Ecosystem Goals

- Achievable
 - High-performance, validated, fully-compliant drivers available in the ecosystem with minimal license hassles (i.e. GPL, BSD, whatever license is appropriate)
- Not Achievable
 - Anticipate every product's unique needs
 - Get driver integrated into OS ahead of products
- Result: Strategy must take varied product goals into account

Linux Driver Flow



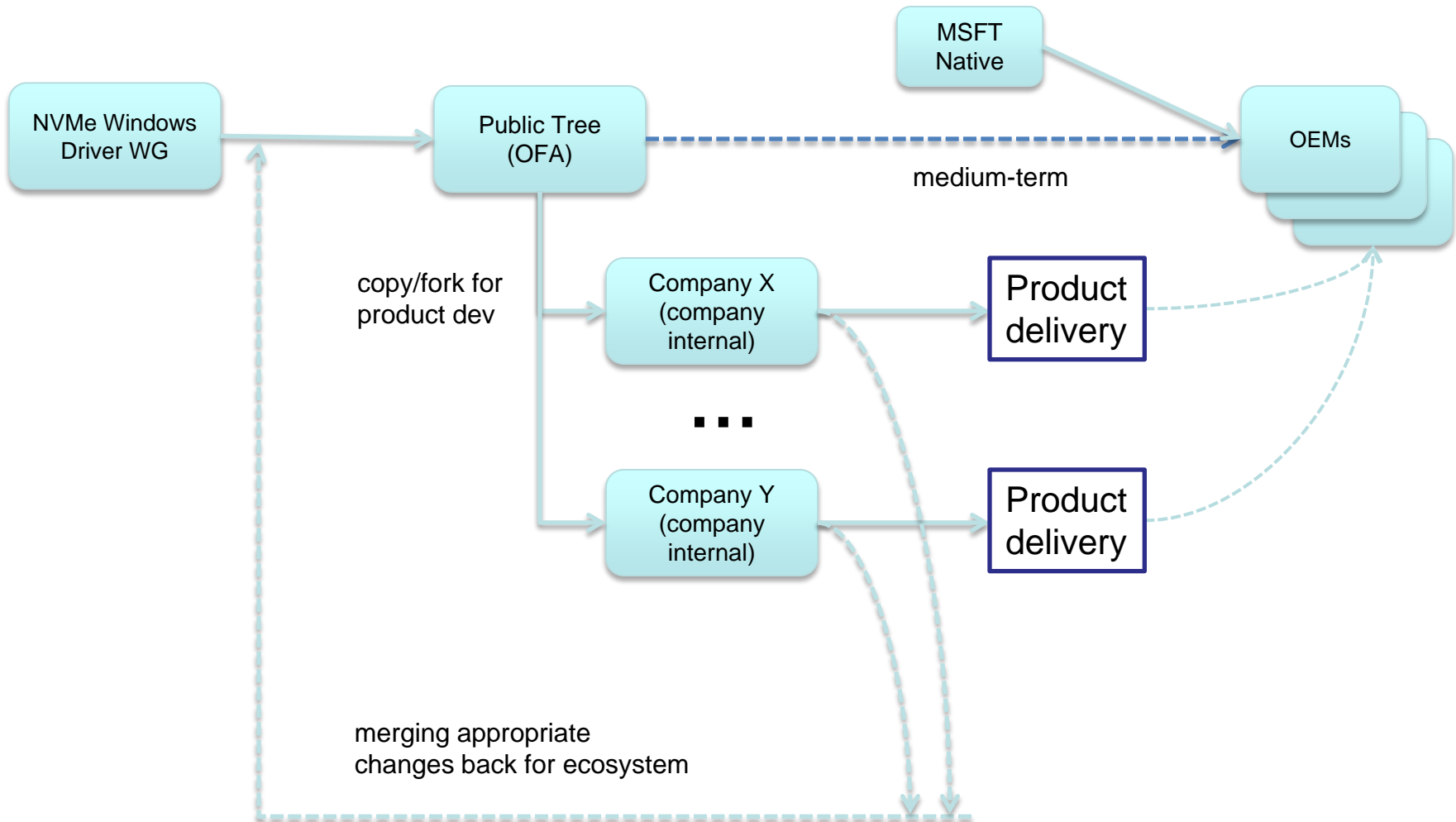
Linux Driver Current Status

- Supports most of the 'M' NVMe features
 - All by production driver
- Supports some of the 'O' NVMe features
- Conservative quality schedule:
 - Beta quality in Q4 '12
 - Production quality in Q2 '13
- Quite likely emerging products will cause validation to happen earlier

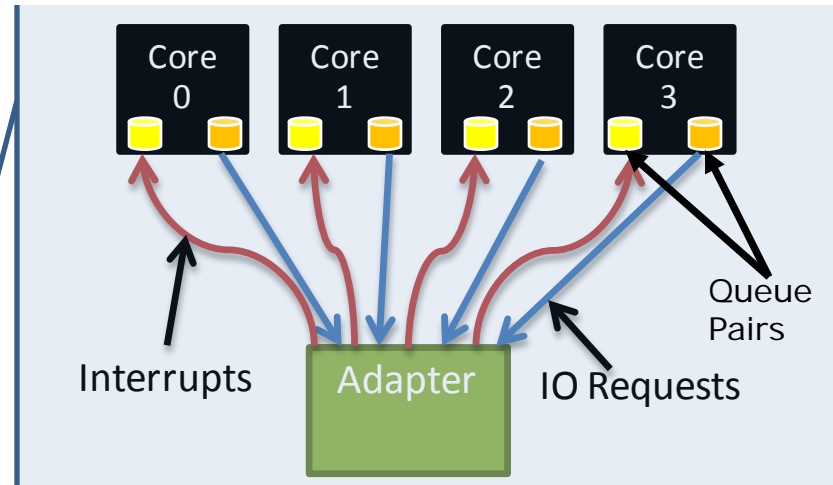
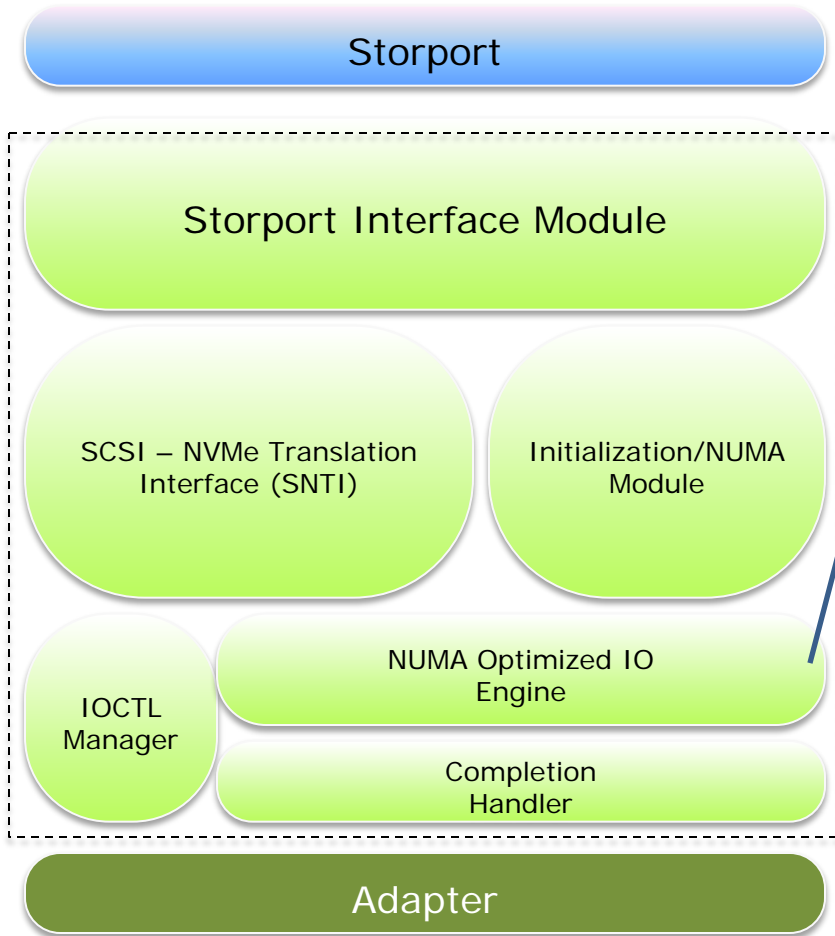
NVMe Linux Patch Submission

- **Understand the Linux kernel patch submission guide and coding style**
 - <http://www.kernel.org/doc/Documentation/SubmittingPatches>
 - <http://www.kernel.org/doc/Documentation/CodingStyle>
- **Clone nvme repo on infradead with 'git clone'**
- **Create a branch and develop your patch**
- **Test your patch against the kernel version used from the cloned repo**
- **Once satisfied your patch is correct, format the patch with 'git format-patch'**
- **Submit your patch to the linux-nvme mailing list using 'git send-email'**
- **The maintainer on the mailing list will either accept your patch or provide feedback.**
 - If accepted, you're done
 - If changes are required, repeat the patch creation/submission process
 - Feel free to post additional questions on the thread that the patch submission created

Windows Driver Flow



NVMe Miniport Architecture



Taking advantage of NVM Express

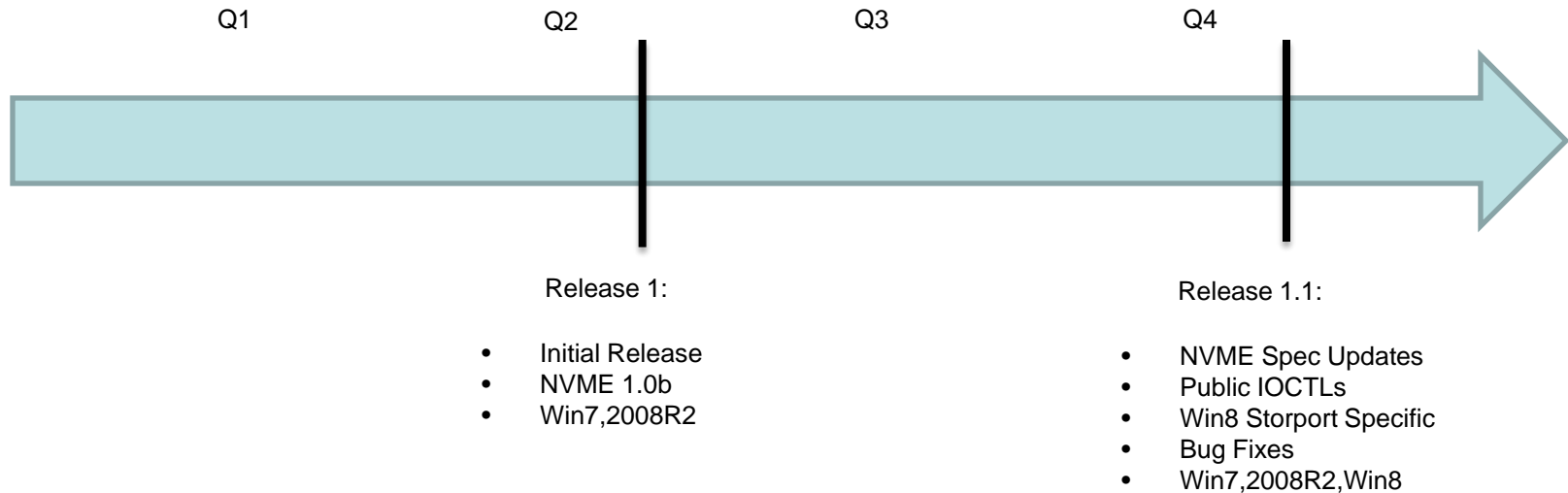
- Distribute I/O Submission & Completion Queues amongst cores
- Submission and Completion Queue memory allocation optimized for NUMA. Queue and MSIX mapping and allocation done during initialization.
- Specify MSI-X vector when creating Completion Queue to process completion optimal core.



Windows Driver Current Status

- Supports all of the 'M' NVMe features
- Supports some of the 'O' NVMe features
- Conservative quality schedule:
 - Beta quality in Q4 '12
 - Production quality in Q2 '13
- Quite likely emerging products will cause validation to happen earlier

OFA NVMe Driver 2012 Release Plans



Releases are binaries for x86, x64, matching source available on OFS via SVN.



Windows Release/Patch Criteria

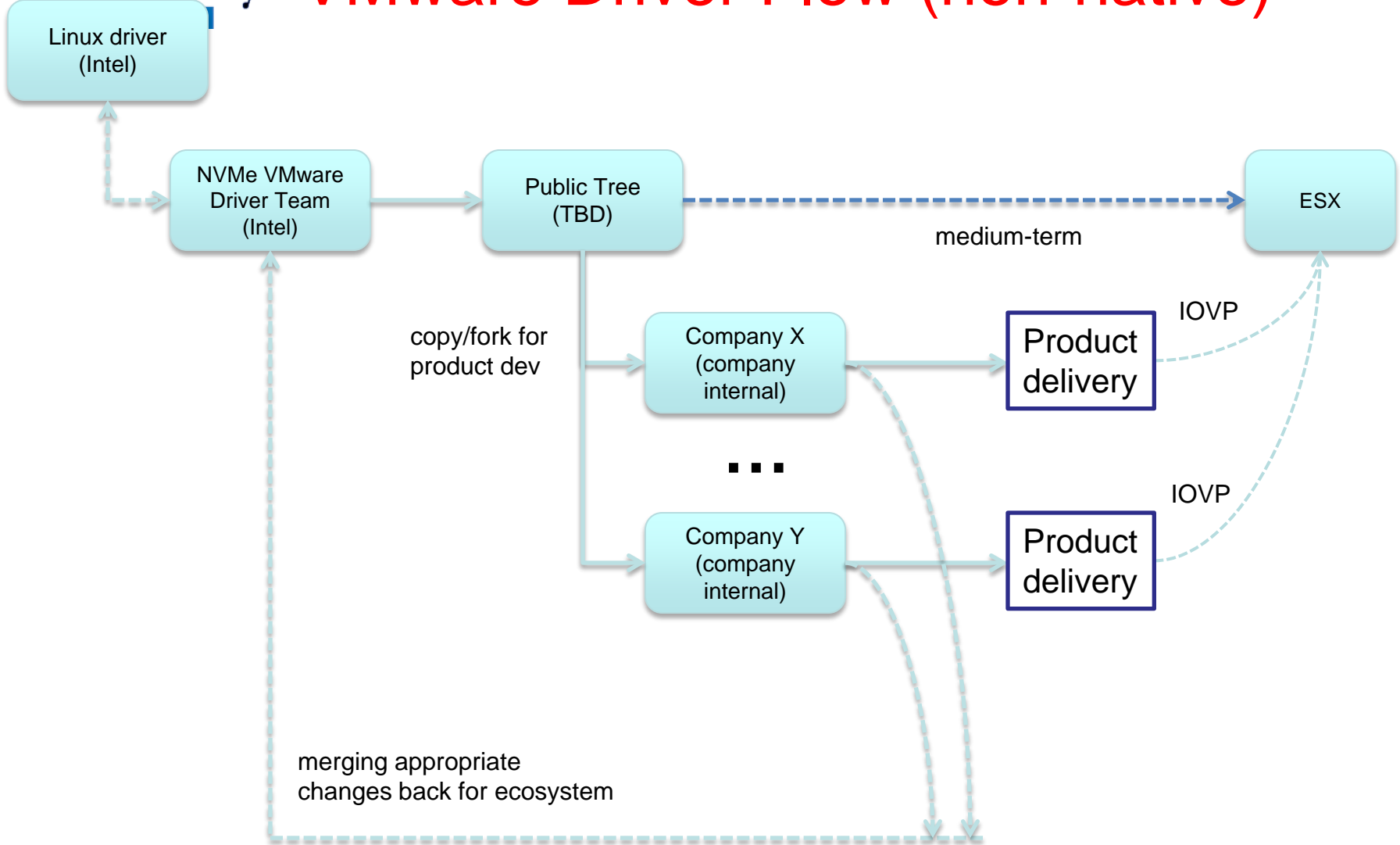
• Reviews

- Patches submitted by anyone, email to distribution list
- Patch submission should include time sensitivity/expectations
- Patch submission should include justification for the patch (what value will it add, and are tradeoffs what are they and why would we want to take a hit). If multiple implementation options were considered, what data/reasoning was behind the implementation choice.
- At a minimum reviews need to be competed by Intel, IDT and LSI representative
- Reviews include compliance with coding guidelines (in SVN) as well as logic

• Testing

- All patches and release candidates required, at a minimum, the following;
 - 1 hour of data integrity testing using sdstress (Microsoft Tool)
 - 1 hour of heavy stress testing using IOMETER covering, at least, 512B, 4KB and 128KB ranging from 1 OIO to 64 OIO both sequential and random
 - Quick and slow format of both MBR and GPT partitioning
 - Microsoft SCSI Compliance, no failures except (warnings OK):
 - READ_CAP due to the test support of a lower SBC than we do
 - WRITE(10) due to what appears to be a false positive (investigating)
 - Performance regression (scripts and procedures) are TBD; will be added shortly.
- Additional testing with other tools is encouraged
- Occurs in all supported OSs for the release
- Minimum test platform is latest QEMU. Those with their HW should test on it as well. QEMU is available at <https://github.com/nvmeqemu/nvmeqemu>

VMware Driver Flow (non-native)



VMware Driver Current Status

- Initial “vmklinux” based driver tracking Linux driver day-to-day
- Native NVMe Driver with pluggable Vendor Extensions planned for future
- VMware’s IOVP program includes workflow for bugs/issues



NVMe Solaris Driver

- Feature Plan
 - Fully implement and conform to 1.0c spec
 - Efficient block interfaces bypassing complex SCSI code path
 - NUMA optimized queue/interrupt allocation
 - Reliable with error detect and recovery fitting into Solaris FMA.
 - Build ZFS with multiple sector size (512, 1K, 2K, 4K) on namespaces.
 - Fit into all Solaris disk utilities and fwflash(1M) for firmware.
 - Boot & install on SPARC and X86.
 - Surprise removal support
 - New Spec: Multipath, SRIOV, SGL, SOP command set, etc.



NVMe Solaris Driver

- Status
 - Have working driver prototype
 - Plan to validate driver against Oracle SSD partners
 - Plan to integrate into S12 and a future S11 Update Release



NVMe UEFI Driver

- Under Development
- Will be Open Sourcing the Driver Around Q1 '13
 - Will include a bug/patch process
- Expect Beta Quality by Q1 '13
- Expect Production Quality by Q2 '13

Strategy Summary

- “Fork and Merge”
 - Maximize re-use, enable continuous improvement of ecosystem code base
 - Allow product groups to focus on their delivery goals
 - Drivers delivered with products should have unique binary names
 - Drivers delivered with products should bind to those products only
 - Support product groups
 - Maintainers available to review driver changes
 - Maintainers maximize ability for a change to be general, flexible for eventual inclusion in ecosystem driver
 - Ask product group to merge changes back when appropriate
 - Not time critical like product delivery
 - NVMe WG encourages this to make it happen