The Nitro Project – Next Generation AWS Infrastructure

Sr. Principal Engineer, EC2 Nitro, AWS
Anthony Liguori <aliguori@amazon.com>

Agenda

Nitro Overview

Evolution of Nitro

Nitro Security Chip Deep Dive

AWS Outposts

After ten years of Amazon Elastic Compute Cloud (Amazon EC2), if we applied all of our learnings, what would a hypervisor look like?

Nitro: Two years later

AWS Nitro

Launched in November 2017

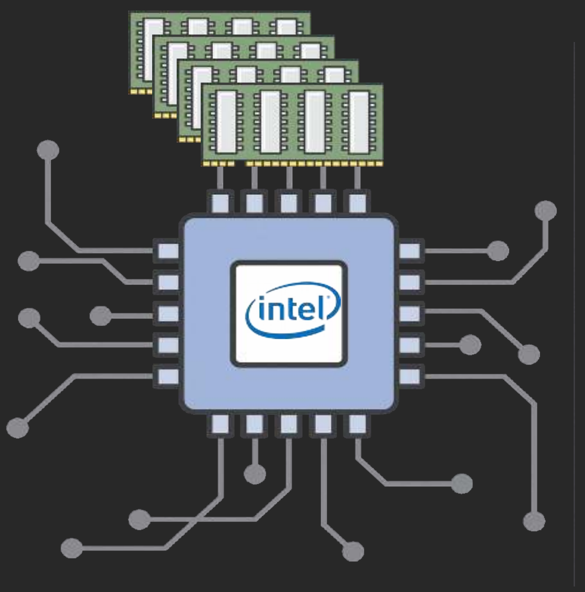
In development since 2013

All new launches use Nitro

Purpose-built hardware/software

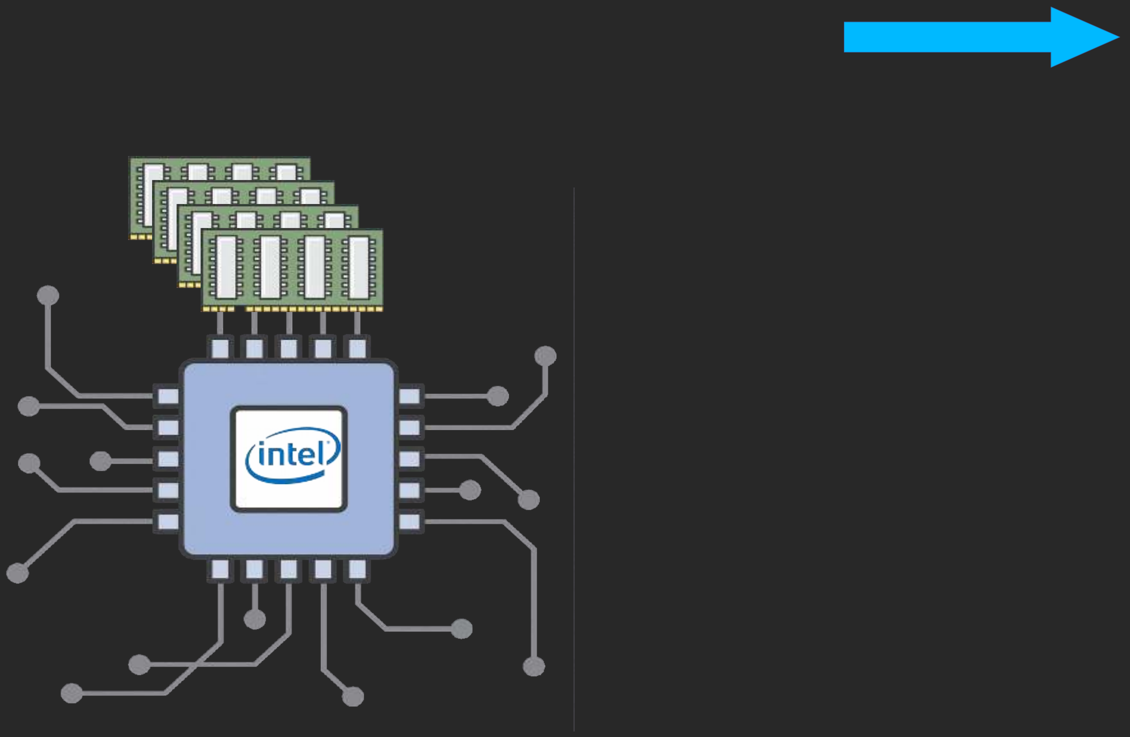
Hypervisor built for AWS

Virtualization



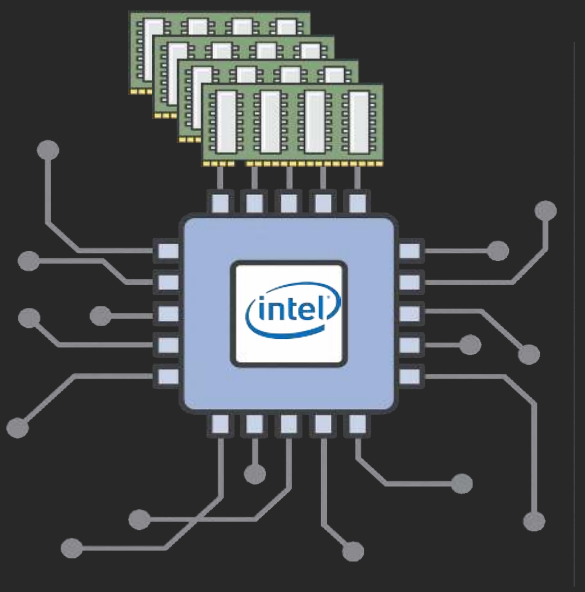
```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

Virtualization



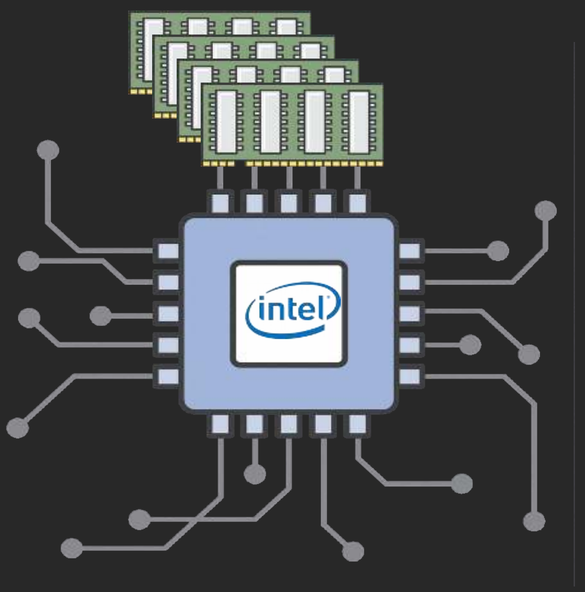
```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

Virtualization



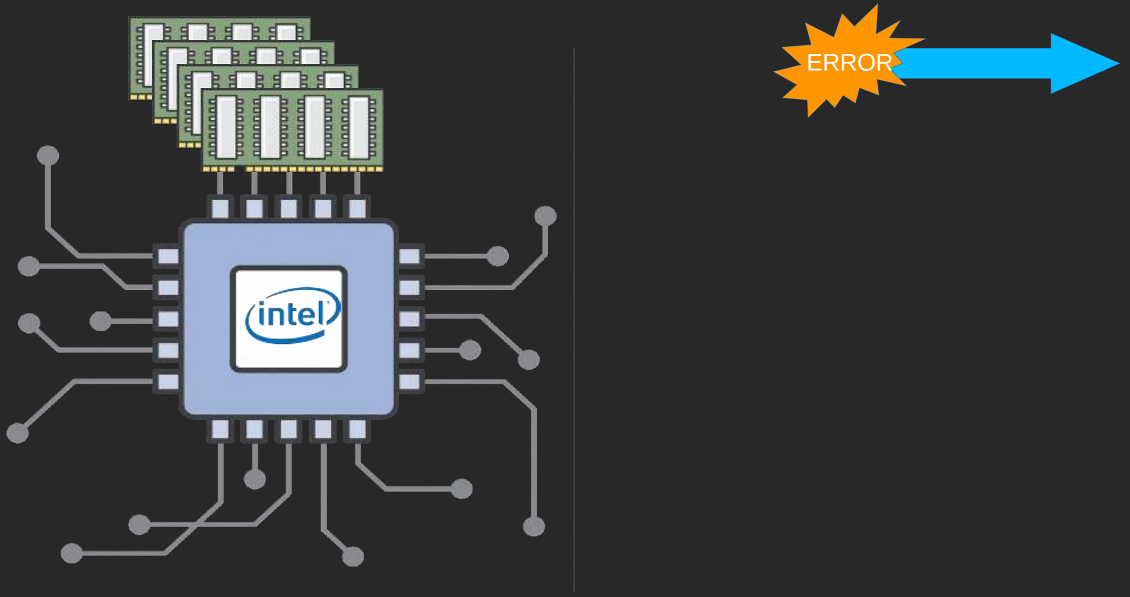
```
<_start>:
        e9 59 e1 17 00          jmpq   ffff82d08037e15e
        0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
        02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
        00 00                   add    %al,(%rax)
        fb                      sti
        4f 52                   rex.WRXB push %r10
        e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
        0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
        d6                      (bad)
        50                      push   %rax
        52                      push   %rdx
        e8 00 00 00 00          callq  ffff82d080200020
        88 00                   mov    %al,(%rax)
```

Virtualization



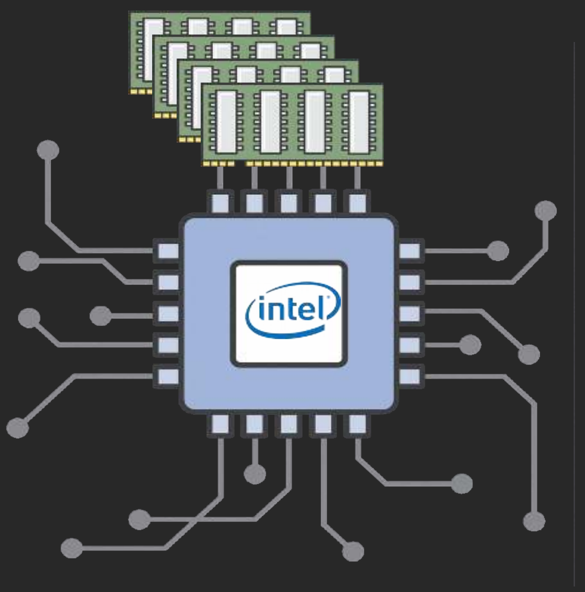
```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

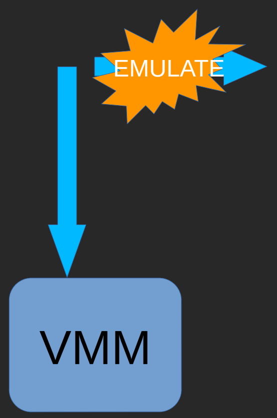
Virtualization



```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

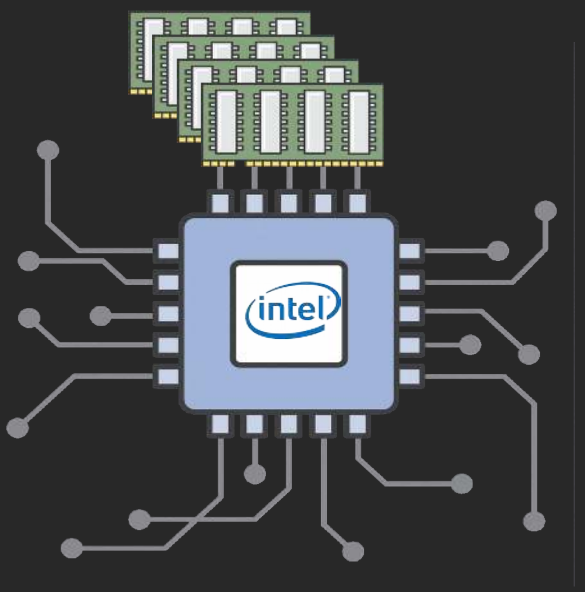
Virtualization





```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

Virtualization

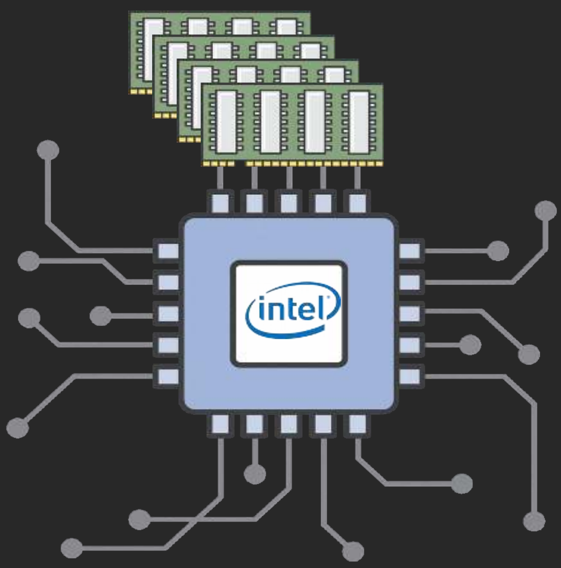


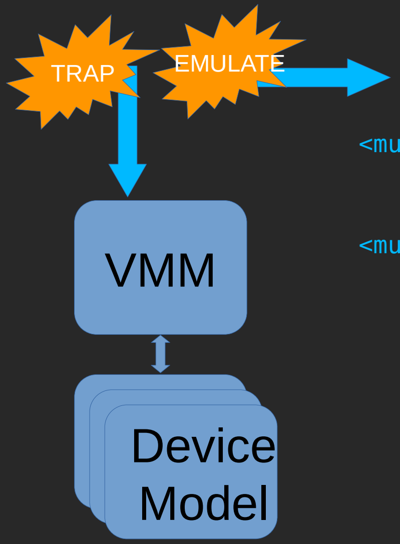
```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

What happened?

- The VMM is the heart of a hypervisor.
- As long as a statistical majority of instructions execute natively, we call this virtualization.
- Not all emulation can be handled by the VMM.

Virtualization





```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   in     $0xf,%al

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

What happened?

- A hypervisor consists of:
- - Virtual Machine Monitor
- - Many device models (10 to 100s)
- - Scheduler, memory manager, etc.

- This was state of the art in 1974

- Not all of the assumptions held true though…

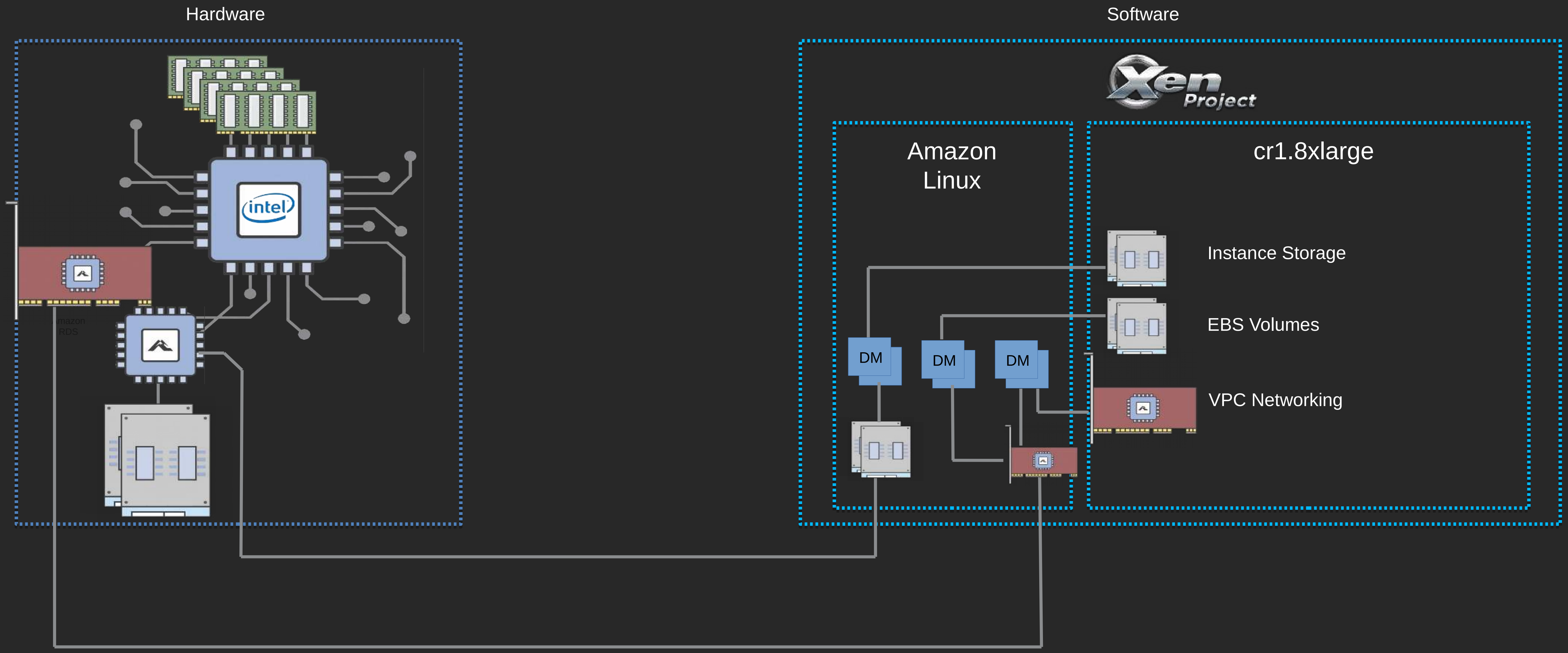
From 1974 to 2006

- Early Intel processors did not trap
- The Xen project found a clever solution
- Paravirtualization modifies the OS to trap
- Hypercalls directly invoke the VMM
- EC2 launched using Xen Paravirtualization

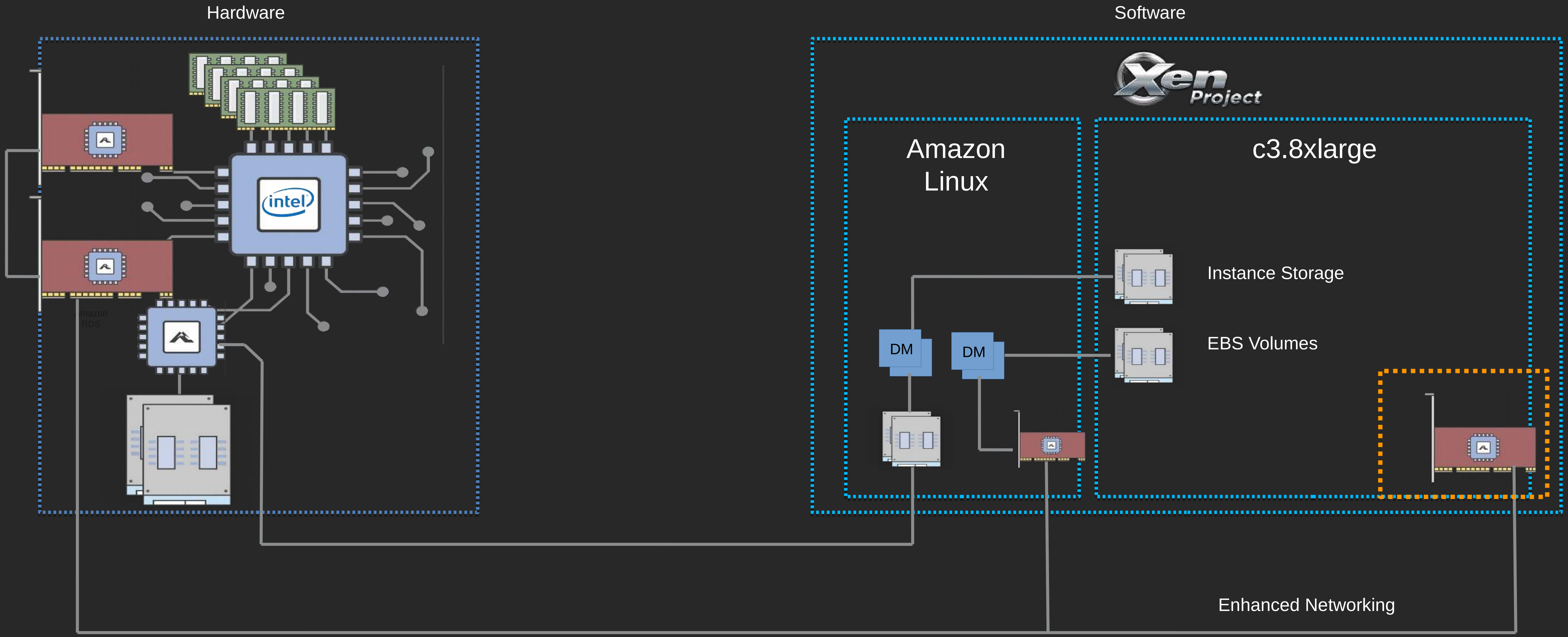
```
<_start>:
       e9 59 e1 17 00          jmpq   ffff82d08037e15e
       0f 1f 00                nopl   (%rax)

<multiboot1_header_start>:
       02 b0 ad 1b 03 00       add    0x31bad(%rax),%dh
       00 00                   add    %al,(%rax)
       fb                      sti
       4f 52                   rex.WRXB push %r10
       e4 0f                   HYPERCALL io_in

<multiboot1_header_end>:
       0f 1f 40 00             nopl   0x0(%rax)

<multiboot2_header_start>:
       d6                      (bad)
       50                      push   %rax
       52                      push   %rdx
       e8 00 00 00 00          callq  ffff82d080200020
       88 00                   mov    %al,(%rax)
```

Evolution of the Nitro System

CR1 (no Nitro) Jan 2013

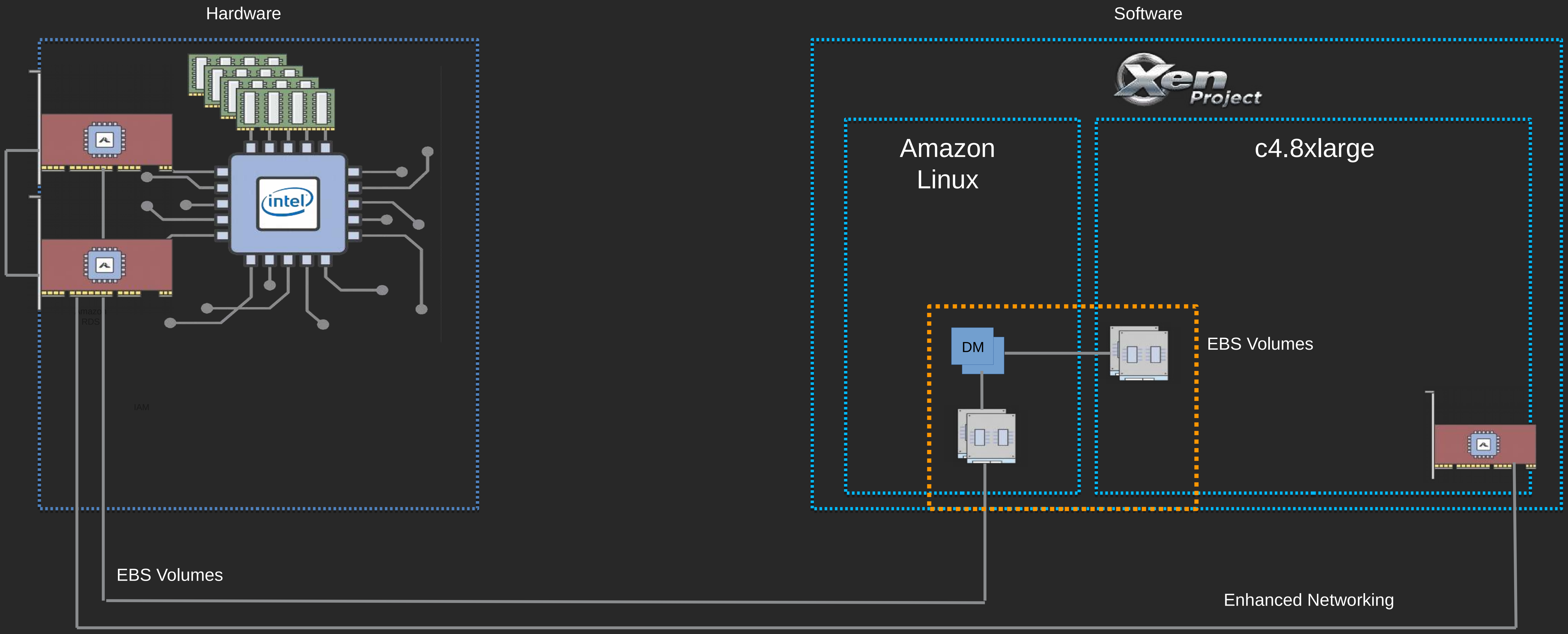


C3 (early Nitro) Nov 2013



C4

Jan 2015



Hardware

Software

Amazon Linux

c4.8xlarge

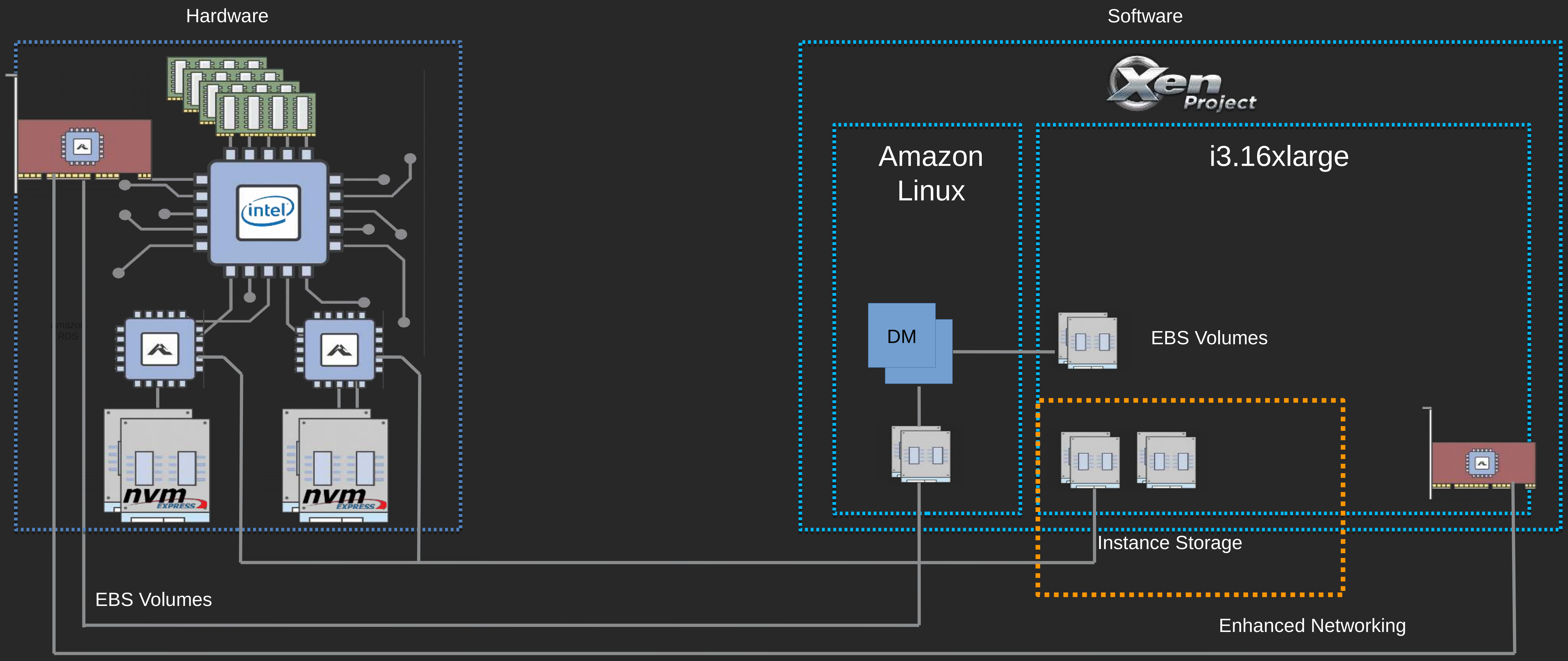
DM

EBS Volumes

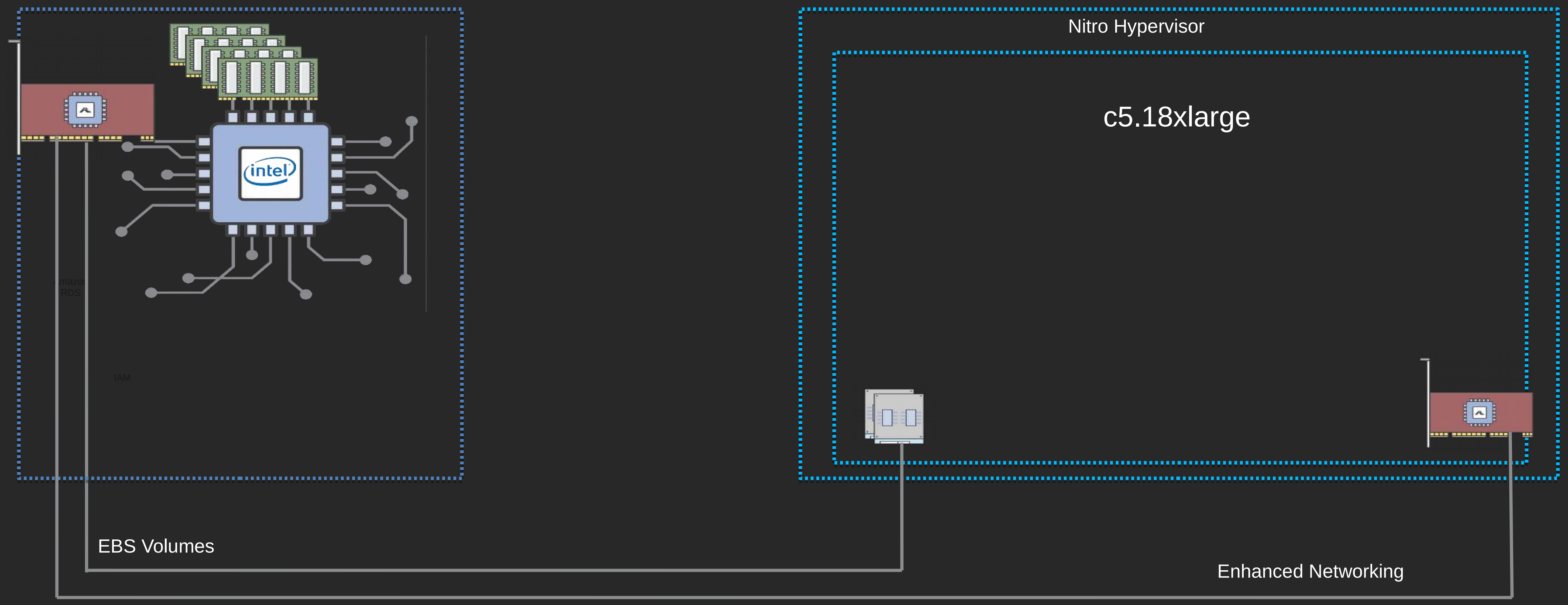
EBS Volumes

Enhanced Networking

I3 Feb 2017


Hardware
Software
Xen Project
Amazon Linux
i3.16xlarge
DM
EBS Volumes
Instance Storage
EBS Volumes
Enhanced Networking

C5 Nov 2017


Hardware
Software
Nitro Hypervisor
c5.18xlarge
intel
mazon
RDS
IAM
EBS Volumes
Enhanced Networking

Nitro in three parts

Nitro Cards

VPC Networking

Amazon Elastic Block Store (Amazon EBS)

Instance Storage

System Controller

Nitro Security Chip

Integrated into motherboard

Protects hardware resources

Hardware Root of Trust

Nitro Hypervisor

Lightweight hypervisor

Memory and CPU allocation

Bare Metal-like performance

Nitro Cards

ENA PCIe Controller

VPC Data Plane

Amazon VPC

NVMe PCIe Controller

EBS Data Plane

Amazon
Elastic Block
Store

NVMe PCIe Controller

Transparent Encryption

Instance
Storage

System Control

Root of Trust

Nitro
Contro
l

Nitro Card for VPC

Amazon VPC

ENA Controller

- Drivers available for all major operating systems
- Independent of fabric

VPC Data Plane

- Encapsulation
- Security Groups
- Limiters
- Routing

Nitro Card for EBS

Amazon Elastic Block Store

NVMe Controller

- Standard drivers broadly available

EBS Data Plane

- Encryption support
- NVM to remote storage protocol

Nitro Card for Instance Storage

Instance Storage

NVMe Controller

- Standard drivers broadly available

Instance Storage Data Plane

- Transparent Encryption
- Limiters
- Drive monitoring

Nitro Card Controller

Nitro
Controller

System Control

- Provides passive API endpoint
- Coordinates all other Nitro Cards
- Coordinates with Nitro Hypervisor
- Coordinates with Nitro Security Chip

Hardware Root of Trust

- Provides measurement and attestation

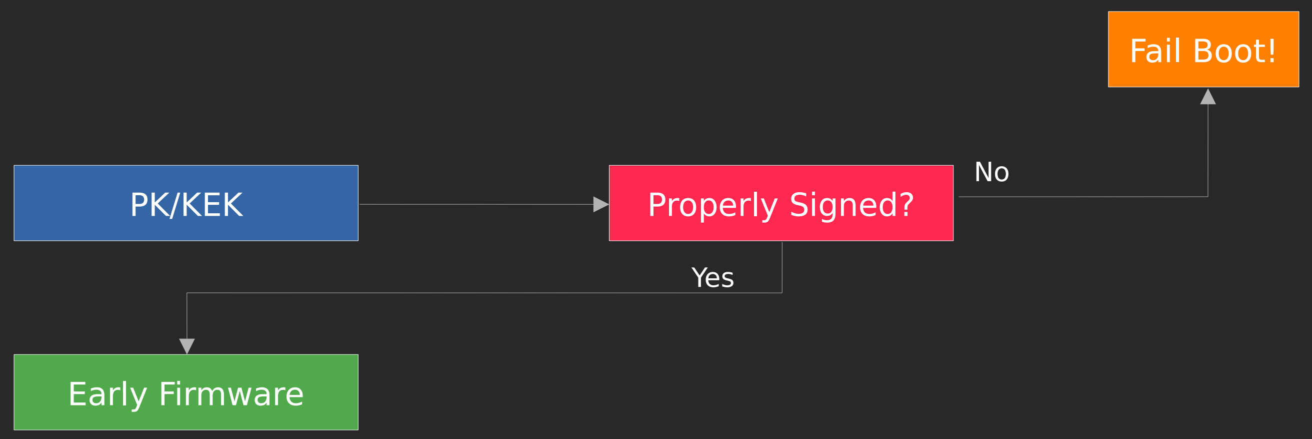
Nitro Security Chip

Custom microcontroller that traps all I/O to non-volatile storage

Controllable from the Nitro Controller to hold system boot

Provides a simple, hardware-based root of trust

UEFI Secure Boot

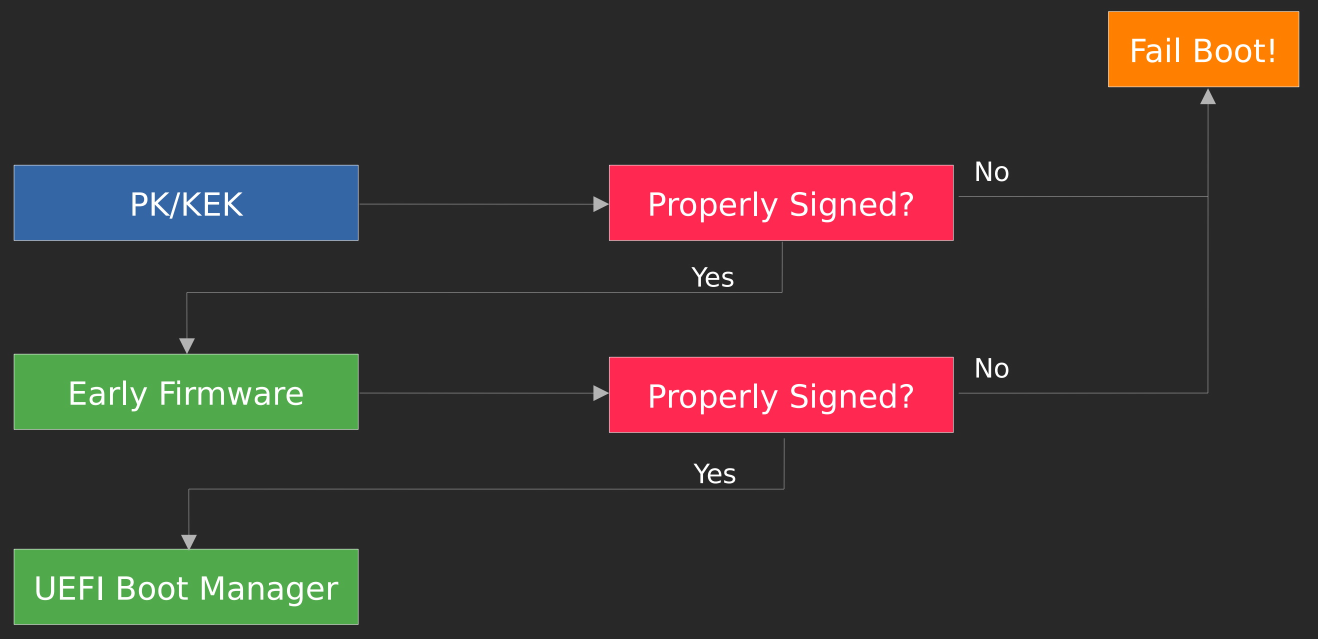


Boot starts untrusted and must prove that system is trustworthy.

Deep complexity with millions of lines of code.

Unavoidable complexity due to need to support legacy and general purpose workloads.

UEFI Secure Boot



Boot starts untrusted and must prove that system is trustworthy.

Deep complexity with millions of lines of code.

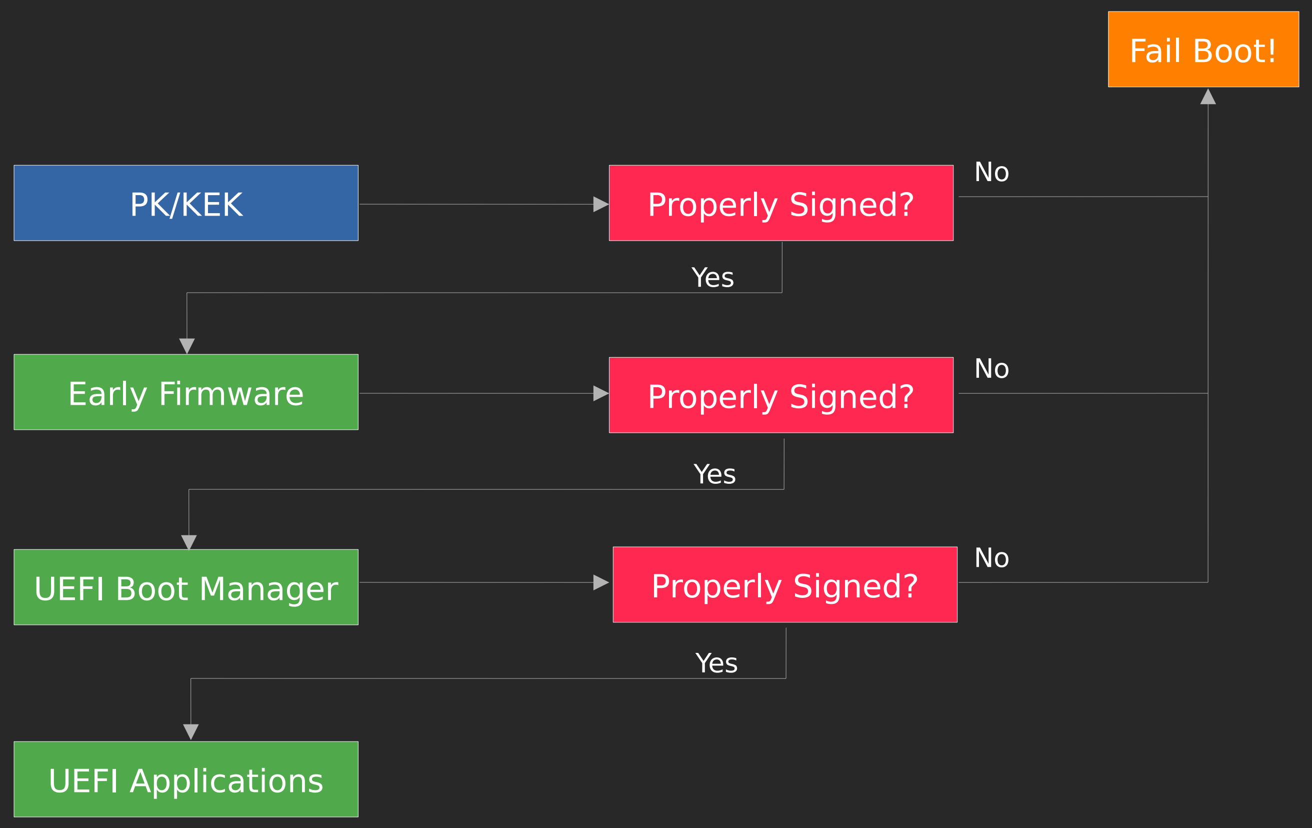
Unavoidable complexity due to need to support legacy and general purpose workloads.

UEFI Secure Boot

Boot starts untrusted and must prove that system is trustworthy.

Deep complexity with millions of lines of code.

Unavoidable complexity due to need to support legacy and general purpose workloads.

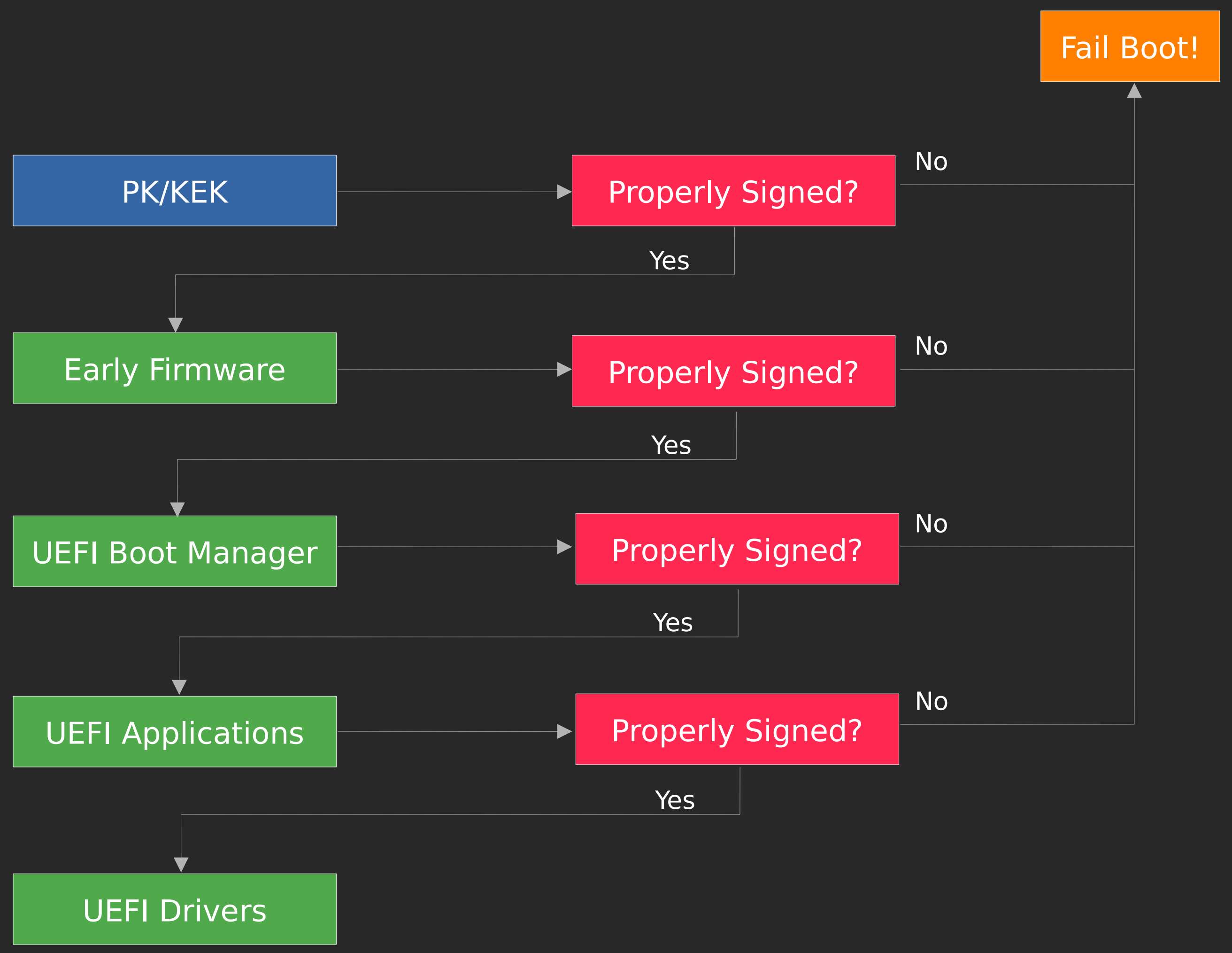


UEFI Secure Boot

Boot starts untrusted and must prove that system is trustworthy.

Deep complexity with millions of lines of code.

Unavoidable complexity due to need to support legacy and general purpose workloads.

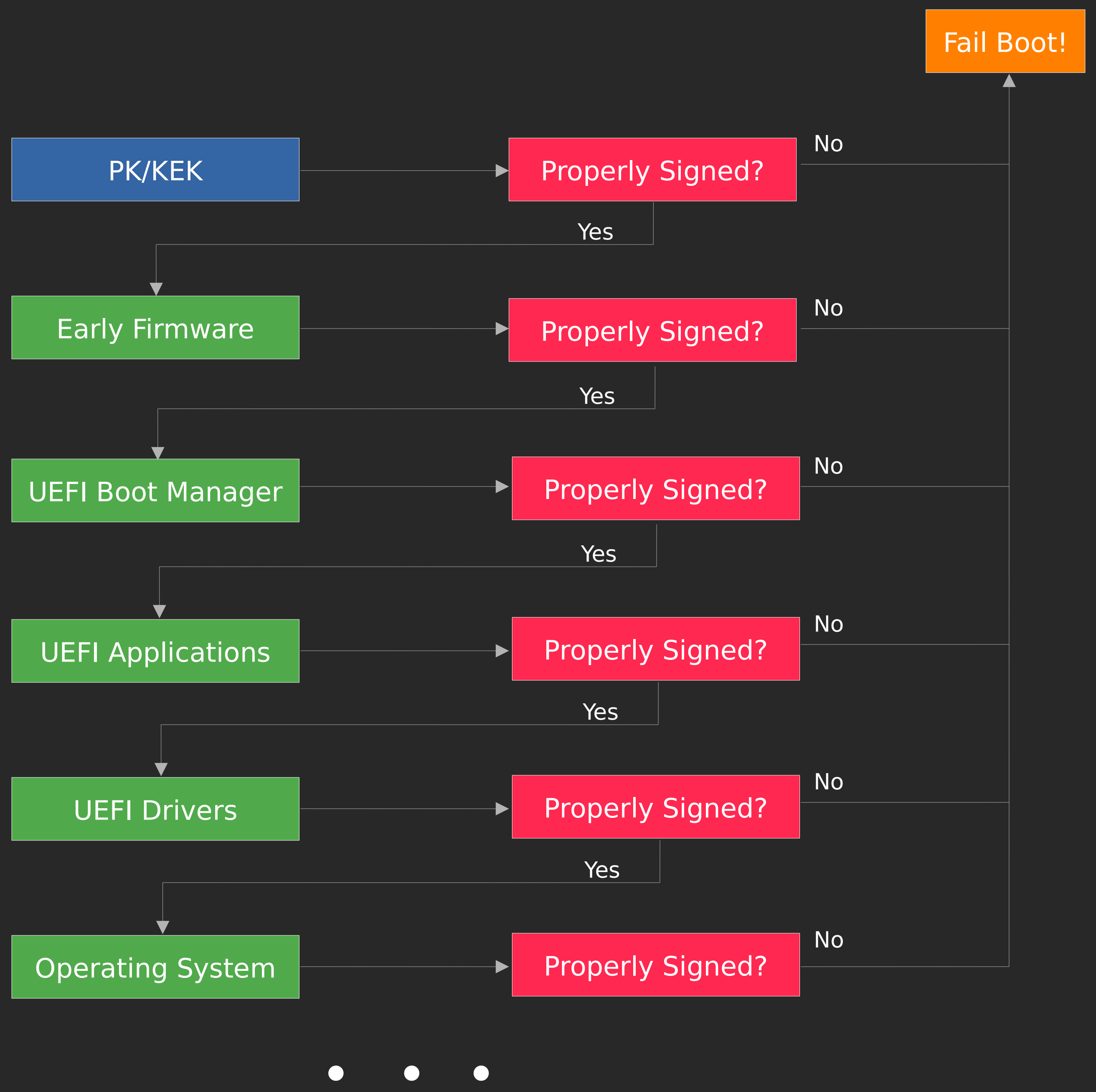


UEFI Secure Boot

Boot starts untrusted and must prove that system is trustworthy.

Deep complexity with millions of lines of code.

Unavoidable complexity due to need to support legacy and general purpose workloads.

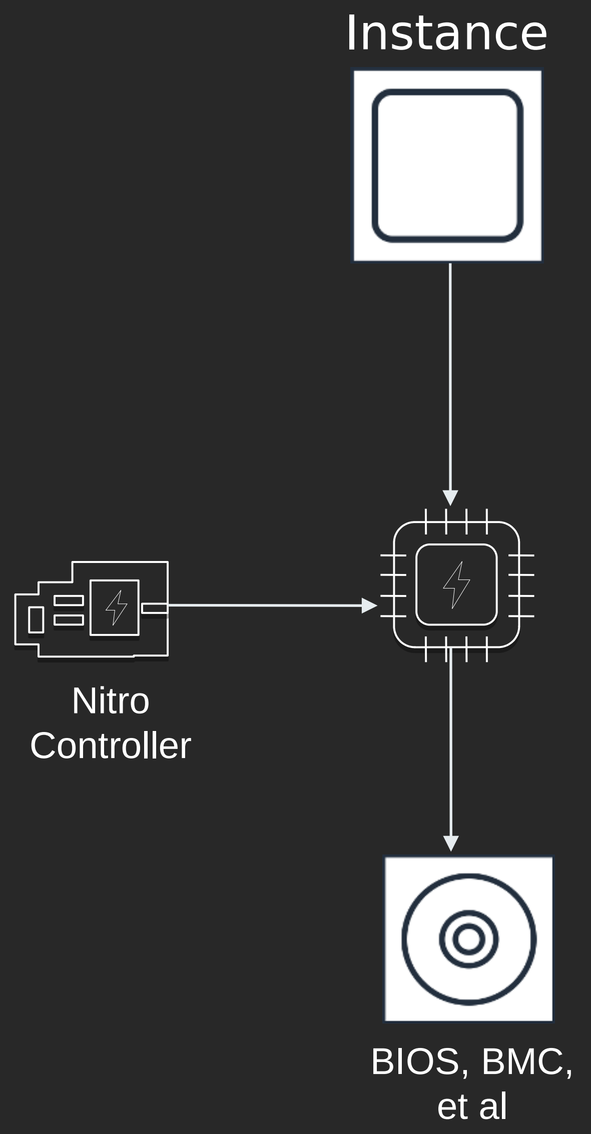


Nitro Hardware Root of Trust

Radical simplification enabled by Nitro Cards.

All write access to non-volatile storage is blocked in hardware.

Simple to understand security due to lack of legacy.

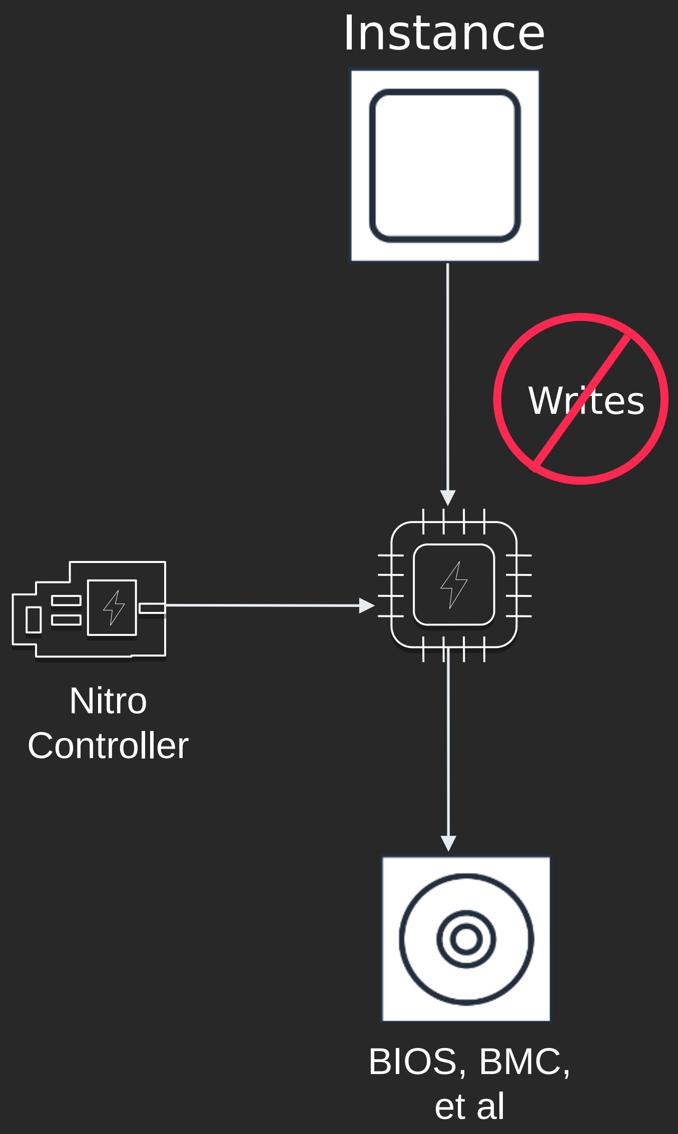


Nitro Hardware Root of Trust

Radical simplification enabled by Nitro Cards.

All write access to non-volatile storage is blocked in hardware.

Simple to understand security due to lack of legacy.



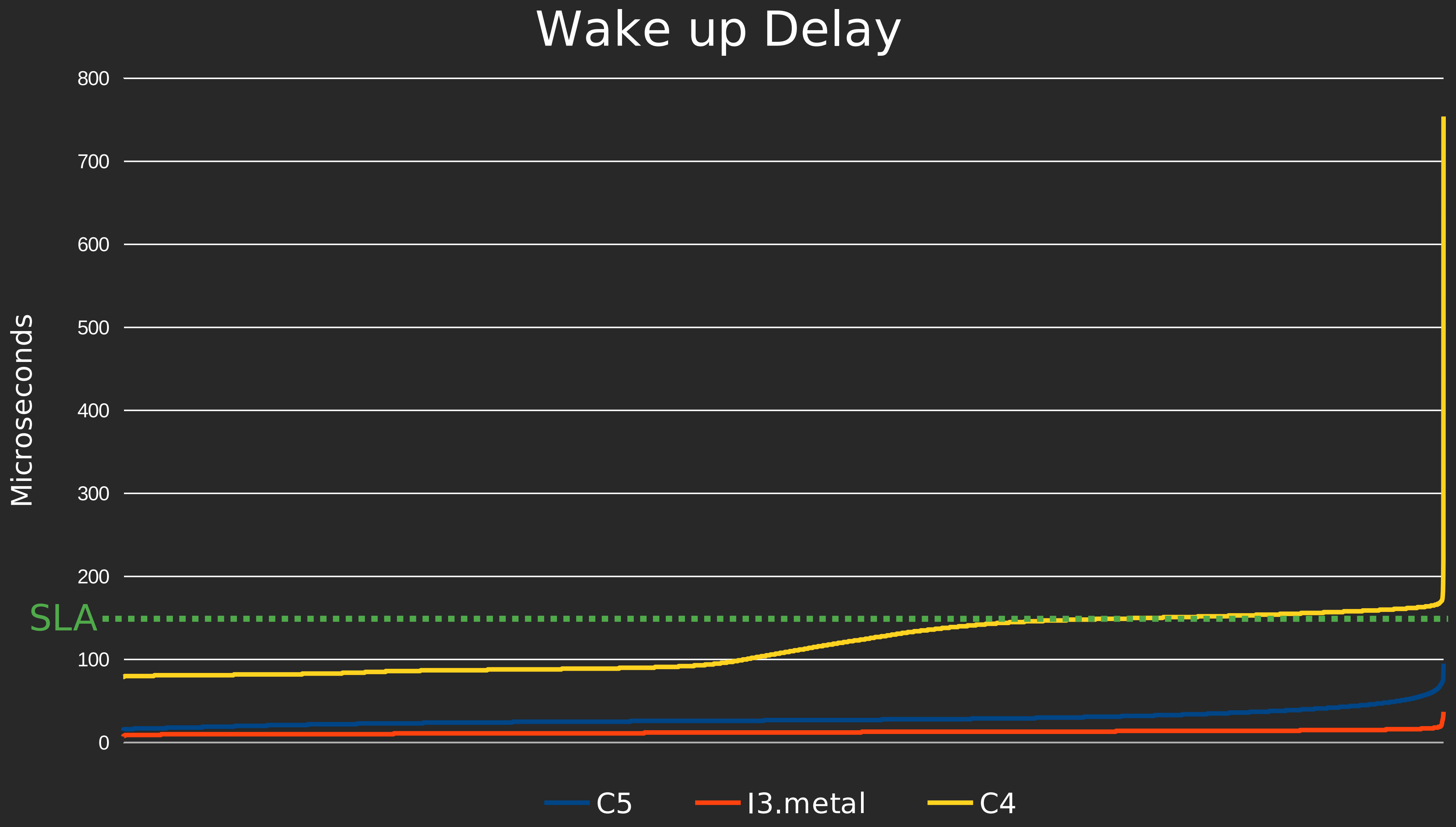
Nitro Hypervisor

KVM-based hypervisor with custom MM and small userspace

Only executes on behalf of instance, quiescent.

With Nitro, the hypervisor can be fast and simple

Nitro Hypervisor Jitter



What comes next?

Nitro: Anywhere you need it

AWS Outposts

Nitro hardware and software in your data center

Access via standard AWS API and console

Deploy apps to Outposts using AWS services