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

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 0d 1b 03 00    add  0x31bad(%rax),%dh
    00 00                add  %al,(%rax)
    fb                    sti
    4f 52                rex.WRXB push %r10
    ed 0f                in  %0xf,%al

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

<multiboot2_header_start>:
    d6                      (bad)
    50                    push  %rax
    52                    push  %rdx
    e8 00 00 00 00        callq  ffff82d080200020
    80 00            mov  %al,(%rax)
```

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Virtualization

```
<start>:
  e9 50 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  ffff82d0000200020
  80 00                   mov    %al,(%rax)
```
Virtualization

```
<start>:
e9 50 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)
1b fb                   sti
4f 52                   rex.WRXB push %r10
ed 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
80 00                   mov  %al,(%rax)
```
Virtualization

```
<start>:
  e9 50 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.XRXB 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 ffff82d000200020
  88 00                   mov %al,(%rax)
```

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Virtualization

```
<_start>:
e9 50 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)
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  ffff82d0800200020
88 00                   mov    %al,(%rax)
```
Virtualization

VMM

<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  ffff82d000200020
  88 00                   mov    %al,%al

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Virtualization

```
<start>:
  e9 50 e1 17 00  jmpq ffff82d08037e15e
  01 1f 00  nopl (%rax)

<multiboot1_header_start>:
  02 b0 ad 1b 03 00  add 0x31bad(%rax),%dh
  00 00  add %al,(%rax)
  fb
  4f 52  sti
  4f 52  rex.WRXB push %r10
  ed 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 ffff82d0080200020
  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 50 e1 17 00
  0f 1f 00
  0f 1f 00
  jmpq ffff82d08037e15e
  nopl (%rax)

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

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

<multiboot2_header_start>:
  d6
  50
  52
  e4 0f
  in $0xf,%al
  push %rax
  push %rdx
  callq ffff82d080200020
  mov %al,(%rax)
```

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
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
Evolution of the Nitro System
C4

Hardware

Software

Amazon Linux
c4.8xlarge

EBS Volumes

Enhanced Networking

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
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

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Nitro Cards

ENA PCIe Controller
VPC Data Plane

NVMe PCIe Controller
System Control

NVMe PCIe Controller
EBS Data Plane

Transparent Encryption
Root of Trust

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Nitro Card for 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

NVMe Controller
Standard drivers broadly available

EBS Data Plane
Encryption support
NVM to remote storage protocol
Nitro Card for Instance Storage

NVMe Controller
Standard drivers broadly available

Instance Storage Data Plane
Transparent Encryption
Limiters
Drive monitoring
Nitro Card 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

Wake up Delay

Microseconds

SLA

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
What comes next?
Nitro: Anywhere you need it

Nitro hardware and software in your data center

Access via standard AWS API and console

Deploy apps to Outposts using AWS services