Amazon Nitro
Presented by Vansh, Kunlin
Agenda

• Why Nitro
• Nitro Overview
• Nitro Cards
• Nitro Security Chip
• Nitro Hypervisor
Why Nitro?

Before Nitro - AWS EC2 CR1 (2013):

• Traditional software-based virtualization
• Relatively simple virtual machine monitor
• Access to device models through privileged OS
Why Nitro?

Drawbacks

• Reserve CPU cores for Dom0 acceleration
• Device models compete for CPU and system resources
• Dom0 OS is too big and complicated

Apply microservices architecture here?
Use specialized hardware?
Nitro Overview

What is Nitro?
• The underlying platform for AWS next generation of EC2 instances
• A combination of dedicated hardware and lightweight hypervisor

Benefits
• Better performance and price
• Faster innovation
• Enhanced security
## Better Performance and Price

<table>
<thead>
<tr>
<th>AWS Instance</th>
<th>vCPU</th>
<th>RAM (GiB)</th>
<th>CPU:RAM</th>
<th>Price ($/Hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W/ Nitro c5.large</td>
<td>2</td>
<td>4</td>
<td>1:2</td>
<td>0.0085</td>
</tr>
<tr>
<td>W/o Nitro c4.large</td>
<td>2</td>
<td>3.75</td>
<td>1:1.875</td>
<td>0.1</td>
</tr>
<tr>
<td>W/ Nitro c5.4xlarge</td>
<td>16</td>
<td>32</td>
<td>1:2</td>
<td>0.68</td>
</tr>
<tr>
<td>W/o Nitro c4.4xlarge</td>
<td>16</td>
<td>30</td>
<td>1:1.875</td>
<td>0.796</td>
</tr>
<tr>
<td>W/ Nitro c5.24xlarge</td>
<td>96</td>
<td>192</td>
<td>1:2</td>
<td>4.08</td>
</tr>
<tr>
<td>W/o Nitro c4.8xlarge</td>
<td>36</td>
<td>60</td>
<td>1:1.875</td>
<td>1.591</td>
</tr>
</tbody>
</table>

Source: https://aws.amazon.com/ec2/pricing/on-demand/
Faster Innovation - Network

• Example: Networking on HPC
• ENA: Elastic Network Adapter
• EFA: Elastic Fabric Adapter (using Nitro Card)
Faster Innovation - Network

Example: Networking on HPC

Benefits of Nitro

• Over 20x increase in PPS performance (Packet Per Second)
• Reduces instance-to-instance latencies
• Enables 100 Gbps of bandwidth performance
EC2 “instance” host architecture

Where should be first part to optimize?
2012 EC2 “instance” host architecture
2013 EC2 “instance” host architecture

- Customer instances
- Hypervisor
- Management, security, and monitoring
- Networking
- Storage
2017 Introducing Nitro Architecture

Server

~100%

Customer instances

Hypervisor

Nitro

Networking

Storage

Management, security, and monitoring
2018 Nitro enabling bare metal instances

- 2018 Nitro enabling bare metal instances

<table>
<thead>
<tr>
<th>Server</th>
<th>~100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer instances</td>
<td></td>
</tr>
</tbody>
</table>

| Nitro | Networking | Storage | Management, security, and monitoring |
Nitro performance
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
NVMe PCIe Controller
EBS Data Plane
NVMe PCIe Controller
Transparent Encryption
System Control
Nitro Controller
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 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 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.
Integrity: Nitro system

Nitro controller is the root of trust

Nitro controller boots from completely private SSD

Boot process formally verified by AWS Automated Reasoning Group:
https://link.springer.com/chapter/10.1007/978-3-319-96142-2_28

Conducts various integrity checks of Nitro computers
Continues on with mainboard boot
When necessary, secure software updates for all components using secure channels, signed binaries

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Integrity: M5d boot process

Mainboard cannot update firmware
But...
Hold mainboard in reset during power-up
Validate all firmware; if valid, continue
Either inject known-good hypervisor
Or boot customer OS/hypervisor AMI from pseudo-NVMe (EBS) volume
Nitro architecture

Private network

Nitro controller & other Nitro computers

Amazon EBS Instance volumes storage

PCle bus

ENA

Intel mainboard
Nitro architecture

Intel mainboard

Private network

Nitro controller & other Nitro computers

Amazon EBS Instance volumes storage

ENA

PCIe bus

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Nitro architecture

Private network

Nitro controller & other Nitro computers

Amazon EBS Instance volumes storage

ENA

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Amazon EBS attach volume API

User calls Amazon EC2 API endpoint
Internal microservices send command to control plane
Control plane sends command to Nitro controller
Nitro controller sends command to EBS controller
EBS controller sends hot-plug event for PCIe device
NVMe device (emulated) shows up on the bus
Key aspects of Nitro

Software device models replaced by (software-defined) hardware devices on the system bus
Amazon EC2 dynamic system changes modeled as hardware events (e.g., NVMe and ENA hot-plug, ACPI power state changes)
Extension of microservice architecture into hardware

- ENA, NVMe protocols are hardened APIs behind which we can innovate
- Data hiding and service decomposition
- Apply (relatively) cheap hardware acceleration to a range of issues

Software elements are also microservices, all dynamically updatable

- No virtual machine (VM) downtime required even for major updates
Encryption—performance

Hardware acceleration allows for line-rate AES-256 encryption of EBS, instance storage, and network without performance penalty

Instance storage: All data

EBS: Now enforceable (for all types) at the account level

Network: Beginning with N types, all direct inter-N customer traffic

Same VPC and across VPC peering, same region

All at up to 100 GB/s

Cf: Project Lever; VPC x-region peering

Caveats
Encryption—key management

EBS: Volumes have independent lifetimes (plus snapshots); therefore, key management via AWS KMS
Instance storage: Locally generated, used, and deleted (instance lifecycle)

VPC:
- Seed materials regionally generated and managed in AWS KMS
- Seeds distributed, not actual secrets; rotated frequently (previous, current, next)

In all cases, plaintext data keys are cached/used only on Nitro computers
  Protected from “customer workload co-processor”
Passive communications design

Hypervisor awaits commands from Nitro controller
- Sent via trusted communications channel
- Never initiates communications with the controller
- Not connected to the network at all

Nitro controller awaits commands from the external control plane
- Listens on network substrate awaiting encrypted, authenticated API calls
- Never initiates outbound connections

Outbound communications from either layer are a clear sign of compromise and are treated accordingly
In Sum

No Dom0 in Nitro hypervisor—greater simplicity and safety
No SSH or other interactive modes anywhere—no direct human access
All access via 100% AuthN/AuthZ APIs with logging/auditing
—no APIs for memory access
Only the Nitro controller has access to the physical Amazon EC2 network; the mainboard does not
End-to-end Nitro system is developed, deployed, and managed by DevSecOps process
In Sum ...

Big benefits from applying microservice concepts to hardware and full-system design

- Unneeded functionality not present
  - What remains is better-defined and easier to reason about: build/test/validate

Stronger single root of trust and greater separation of concerns (and code, and teams) along every dimension

Nitro building blocks will continue to be applied in Amazon EC2 and beyond
  - Firecracker, Outposts, etc.

Lots of security value already, and there’s more that we can do!
Thank you!