# Building a Future-Proof Cloud Infrastructure

A Unified Architecture for Network, Security, and Storage Services



SILVANO GAI

*With Contributions by*

**Roger Andersson, Diego Crupnicoff,** and **Vipin Jain**

# Building a Future-Proof Cloud Infrastructure

## A Unified Architecture for Network, Security, and Storage Services

Silvano Gai

With Contributions by
Roger Andersson,
Diego Crupnicoff, and Vipin Jain

Addison-Wesley

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2019956931



Pearson Education, Inc.



ISBN-13: 978-0-13-662409-7

ISBN-10: 0-13-662409-X

ScoutAutomatedPrintCode

**Editor-in-Chief**
Mark Taub

**Product Manager**
James Manly

**Managing Editor**
Sandra Schroeder

**Senior Project Editor**
Lori Lyons

**Copy Editor**
Paula Lowell

**Production Manager**
Vaishnavi/codeMantra

**Indexer**
Erika Millen

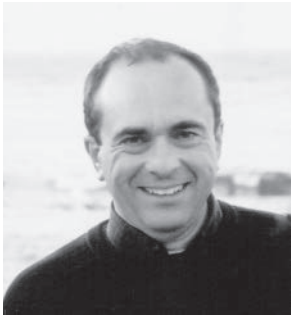
**Proofreader**
Abigail Manheim

**Editorial Assistant**
Cindy Teeters

**Cover Designer**
Chuti Prasertsith

**Compositor**
codeMantra

# About the Authors



**Silvano Gai,** who grew up in a small village near Asti, Italy, has more than 35 years of experience in computer engineering and computer networks. He is the author of several books and technical publications on computer networking as well as multiple Internet Drafts and RFCs. He is responsible for 50 issued patents. His background includes seven years as a full professor of Computer Engineering, tenure track, at Politecnico di Torino, Italy, and seven years as a researcher at the CNR (Italian National Council for Scientific Research). For the past 20 years, he has been in Silicon Valley where, in the position of Cisco Fellow, he was an architect of the Cisco Catalyst family of network switches, of the Cisco MDS family of storage networking switches, of the Nexus family of data center switches, and the Cisco Unified Computing System (UCS). Silvano is currently a Fellow with Pensando Systems.

**Roger Andersson** has spent more than 28 years in the computer and storage industry with work experience that spans across EMC/Data General, Pure Storage, Veritas/Symantec and Nuova Systems/Cisco UCS, focusing on software automation, OS provisioning, and policy-driven management at scale. Roger's roles started in hardware engineering, moved to software engineering, and for the past 16 years has been in technical product management roles. Roger is currently working at Pensando as a Technical Product Manager focusing on Distributed Service Management at scale. Roger was born in Stockholm, Sweden.

**Diego Crupnicoff** has been a Fellow at Pensando Systems since May 2017. Prior to that, Diego served as VP Architecture at Mellanox Technologies where he worked since its inception in 1999, driving chip and system architectures for multiple generations of Ethernet and RDMA products. Diego has been a member of the InfiniBand Trade Association since its early days and took part in the definition of the InfiniBand RDMA Standard. Among other roles, Diego chaired the IBTA Technical Working Group for many years. He was also among the founding directors of the OpenFabrics Alliance and chaired its Technical Advisory Council for several years. Over the past two decades, Diego has participated in multiple other SDOs and Tech Committees, including the IEEE802, IETF, T11, NVME, and the ONF. Diego is an inventor in multiple patents on the areas of computer networks and system architecture. He holds a B.Sc. in Computer Engineering (Summa Cum Laude) and an M.Sc. in EE. (Summa Cum Laude), both from the Technion - Israel Institute of Technology.

**Vipin Jain** is a passionate engineer with 20 years of industry experience. Over the years, he has contributed in the areas of switching, routing, network protocols, embedded systems, ASIC architecture, data path design, distributed systems, software-defined networking, container networking, orchestration systems, application security, open source, cloud infrastructure, and DevOps.

He holds numerous patents and has been a speaker at many conferences, author of IETF RFCs, and been a developer evangelist for his open source work. He enjoys coding for work and for fun. He also enjoys snowboarding, hiking, kayaking, and reading philosophy. He holds a bachelor's degree in Computer Science from NIT Warangal, India. He has worked in multiple successful startups in technical and management leadership roles. He is founder and CTO at Pensando Systems.

Chapter 8

# NIC Evolution

Previous chapters presented the importance of domain-specific hardware in a distributed services platform. One of the places where this domain-specific hardware can be located on is the network interface card ( NIC).

Historically the NIC was designed to satisfy the server's need to communicate over one or more Ethernet networks effectively. In its purest form, a NIC is a device receiving packets arriving from the network and sending them on the server bus and vice versa.

This chapter discusses the NIC evolution from a primitive packet-passing widget of the 1990s to today's SmartNIC—a sophisticated device capable of hosting domain-specific hardware either in the form of an additional ASIC or on the NIC ASIC itself.

An example of additional ASIC implementation is the Microsoft Azure SmartNIC described in section 2.8.1 that has a field programmable gate array (FPGA) on board next to a regular NIC ASIC. An example of extra hardware on the NIC ASIC is the Broadcom BCM58800 that includes, among other things, several ARM cores.

Also, memory considerations are essential because a distributed services platform needs memory to store its multiple tables.

We have seen in Chapter 6, "Distributed Storage and RDMA Services," that locating the domain-specific hardware on the NIC is the preferred embodiment for storage and RDMA services because they are tied to the server memory subsystem and PCIe bus semantics.

When appropriately implemented, the addition of domain-specific hardware to a NIC can reduce the server CPU load caused by network, storage, and security services and, therefore, return precious server CPU cycles to user applications.

# 8.1 Understanding Server Buses

Server buses are in constant evolution, but all modern servers use PCI Express (Peripheral Component Interconnect Express), officially abbreviated as PCIe or PCI-e, a high-speed serial computer expansion bus.

Whereas PCI and PCI-X were bus topologies, PCIe is a point-to-point standard that supports bridges. The resulting topology is a tree structure with a single root complex, as illustrated in Figure 8-1.

The PCIe root complex is responsible for enumeration of PCIe resources and system configuration; it also manages interrupts and errors for the PCIe tree. Historically, the root complex functionality was located in the South Bridge (also known as ICH or I/O Controller Hub), whereas today processors often integrate it onboard.

PCIe is composed of a variable number of "lanes," whose allowed values are 1, 2, 4, 8, 12, 16, or 32 [1]. The notation used is in the form "x4," which means four lanes. Each lane is composed of two differential signaling pairs: one RX pair and one TX pair. Therefore, each lane supports full-duplex operation.

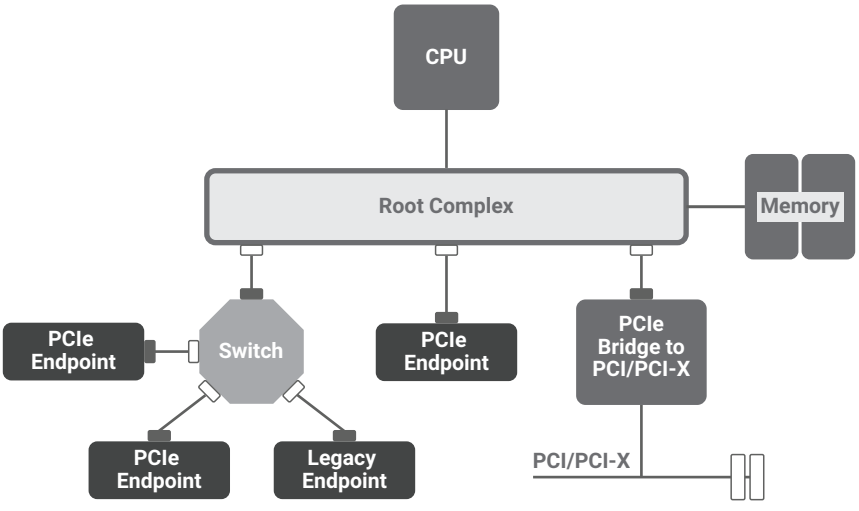



**FIGURE 8-1** PCIe Root Complex

PCIe exists in different "generations" or "versions." Currently, the most commonly used is PCIe Gen 3, and the industry is getting ready to adopt Gen 4. Table 8-1 summarizes the differences. For each version, the throughput is expressed in Giga transfers per second (GT/s), in GB/s (Gigabyte/s), and Gb/s (Gigabit/s). The number of transfers also includes the overhead encoding bits that are excluded in GB/s and Gb/s.

**TABLE 8-1** PCIe Performance

| PCIe Version | Year | Line Code | Transfer Rate | Throughput | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | x1 | x2 | x4 | x8 | x16 |
| 1.0 | 2003 | 8b/10b | 2.5 GT/s | 250 MB/s<br>2 Gb/s | 0.5 GB/s<br>4 Gb/s | 1.0 GB/s<br>8 Gb/s | 2.0 GB/s<br>16 Gb/s | 4.0 GB/s<br>32 Gb/s |
| 2.0 | 2007 | 8b/10b | 5.0 GT/s | 500 MB/s<br>4 Gb/s | 1.0 GB/s<br>8 Gb/s | 2.0 GB/s<br>16 Gb/s | 4.0 GB/s<br>32 Gb/s | 8.0 GB/s<br>64 Gb/s |
| 3.0 | 2010 | 128b/130b | 8.0 GT/s | 985 MB/s<br>7/88 Gb/s | 1.97 GB/s<br>15.76 Gb/s | 3.94 GB/s<br>31.52 Gb/s | 7.88 GB/s<br>63.04 Gb/s | 15.8 GB/s<br>126.08 Gb/s |
| 4.0 | 2017 | 128b/130b | 16.0 GT/s | 1969 MB/s<br>15.76 Gb/s | 3.94 GB/s<br>31.52 Gb/s | 7.88 GB/s<br>63.04 Gb/s | 15.75 GB/s<br>126.08 Gb/s | 31.5 GB/s<br>252.16 Gb/s |
| 5.0 | 2019 | 128b/130b | 32.0 GT/s | 3938 MB/s<br>31.52 Gb/s | 7.88 GB/s<br>63.04 Gb/s | 15.75 GB/s<br>126.08 Gb/s | 31.51 GB/s<br>252.16 Gb/s | 63.0 GB/s<br>54.32 Gb/s |

An error-correcting code protects data transfers. For example, in Gen 3 each lane is capable of transferring 8 Gb/s but, because the encoding requires a transfer of 130 bits to get 128 bits of data, the effective transfer rate is 7,877 Mb/s, equivalent to 984.6 MB/s. Table 8-1 also shows that four Gen 3 lanes support a 25 Gb/s Ethernet connection, eight lanes support 50 Gb/s Ethernet, and sixteen support 100 Gb/s Ethernet.

## 8.2 Comparing NIC Form Factors

NICs can either be soldered directly in the motherboard in an arrangement called LOM (LAN On Motherboard) or can be installed as separate pluggable cards in PCIe slots. These cards have three main form factors: PCIe plugin, proprietary mezzanine, and OCP mezzanine.

### 8.2.1 PCI Plugin Cards

The plugin card is probably the most common form factor. It is used almost universally on rack-mounted servers. These servers have one or more PCIe slots for PCIe cards.

Figure 8-2 shows an Intel plugin card, PCI Express Gen 2, 5 GT/s, x4 Lane; this is an old card shown for historical reasons with 2 x 1 Gbps ports. Intel now makes much faster state-of-the-art cards. Note the absence of a heatsink or fan.

Figure 8-3 shows a Broadcom BCM957454A4540C NeXtreme E-Series Single-Port 1/10/25/40/50/100 Gb/s Ethernet PCI Express Gen 3 x16.

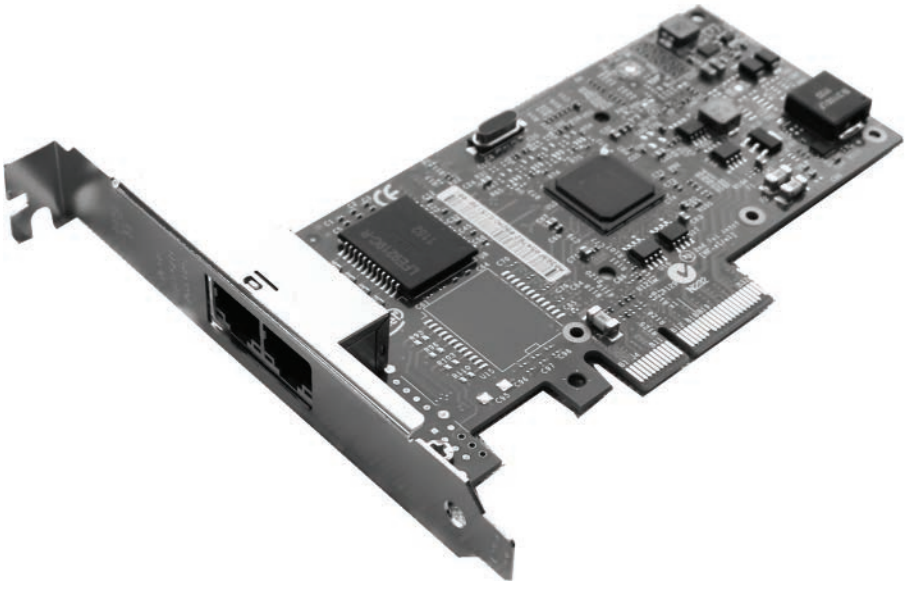


**FIGURE 8-2** Intel I350-T2 Intel Ethernet Server Adapter

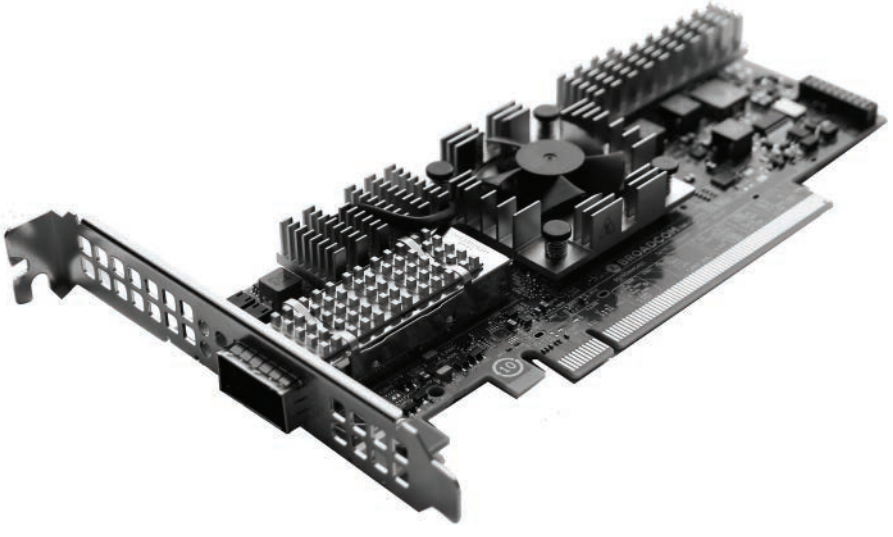


**FIGURE 8-3** Broadcom NeXtreme E-Series

PCIe slots have a standard power and cooling limitation of 25 Watts per slot [2], and this limits the amount of hardware that can be installed on a PCIe board. Some NIC manufacturers are not able to comply with this strict budget requirement and use two adjacent slots for a single card. Some server vendors offer more power per slot, but all the power consumed must be dissipated, increasing operation cost.

In general, the design of domain-specific hardware must take into significant consideration that power is limited and, therefore, use all possible design techniques to reduce power consumption.

### 8.2.2 Proprietary Mezzanine Cards

Many enterprise datacenters use Blade servers as a way to optimize cabling and power efficiency when compared to rack-mounted servers. The form factors of these servers vary with the manufacturer, and they are proprietary.

Blade servers cannot host generic PCIe plugin cards but require "mezzanine daughter cards" that also have proprietary form factors and proprietary connectors. The daughter card is parallel to the main board in a stacking configuration.

Although the connector is proprietary, the protocol spoken on the connector is always PCIe.

Figure 8-4 shows a NC532m 10GbE 2-port adapter for a Hewlett-Packard Enterprise BladeSystem c-Class.

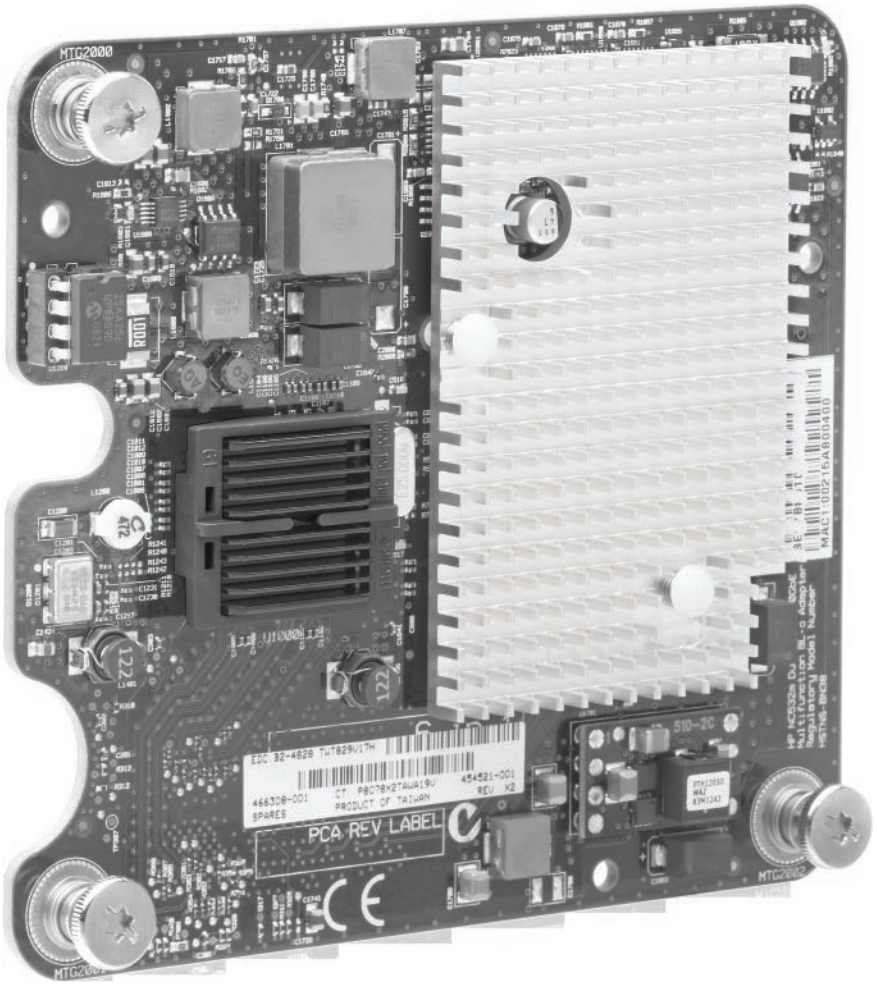



**FIGURE 8-4** HPE Card in a Mezzanine Form Factor (Courtesy of Hewlett Packard Enterprise)

### 8.2.3 OCP Mezzanine Cards

The Open Compute Project (OCP) [3] is a community of engineers whose mission is to standardize hardware designs for server, storage, and datacenter hardware.

Particularly relevant for our discussion is the OCP NIC subgroup currently working on a follow-on to the OCP Mezzanine 2.0 design specification called OCP NIC 3.0, a new specification that supports small form factor cards and large form factor cards. The small card allows for up to 16 PCIe lanes, and the large card supports up to 32 PCIe lanes.

Figure 8-5 shows a pictorial representation of these two cards.

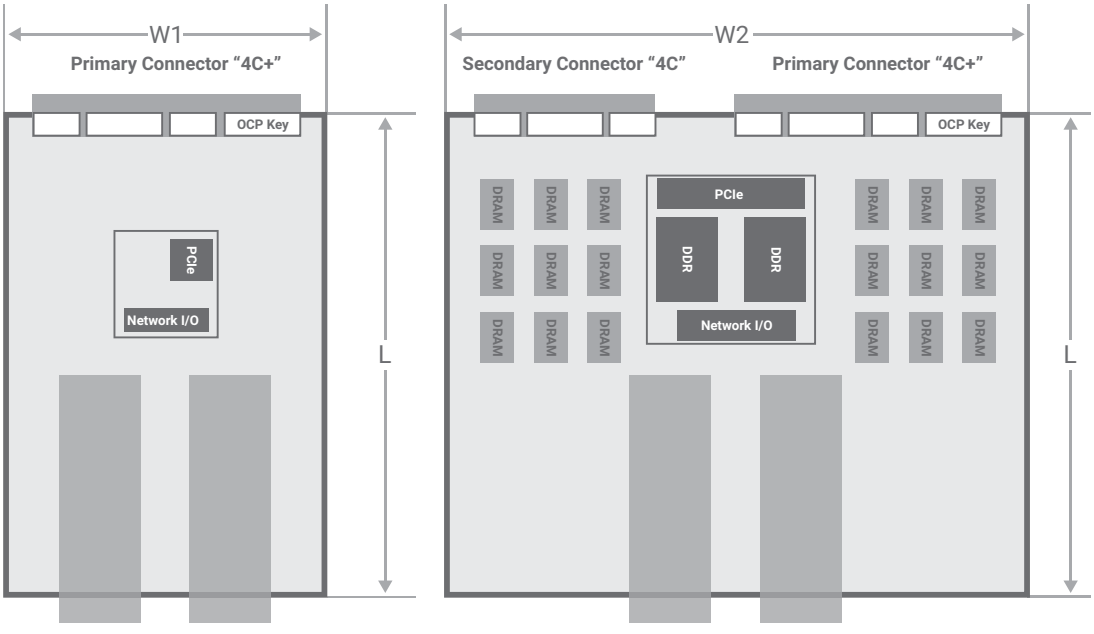



**FIGURE 8-5** OCP 3.0 Small and Large Cards (Courtesy of the Open Compute Project Foundation)

OCP 3.0 cards support up to 80 Watts in a small form factor and up to 150 Watts in large form factor. However, the limiting factor is not current delivery but thermal dissipation. Each platform will limit the power consumption based on its thermal capabilities, and a typical range will be 15 to 25 watts, as in the case of PCIe plugin cards, with some platforms trying to push it somewhere around 35 watts.

Figure 8-6 shows a Mellanox ConnectX-5 Dual Port 100GbE in OCP 3.0 form factor for an OCP server.

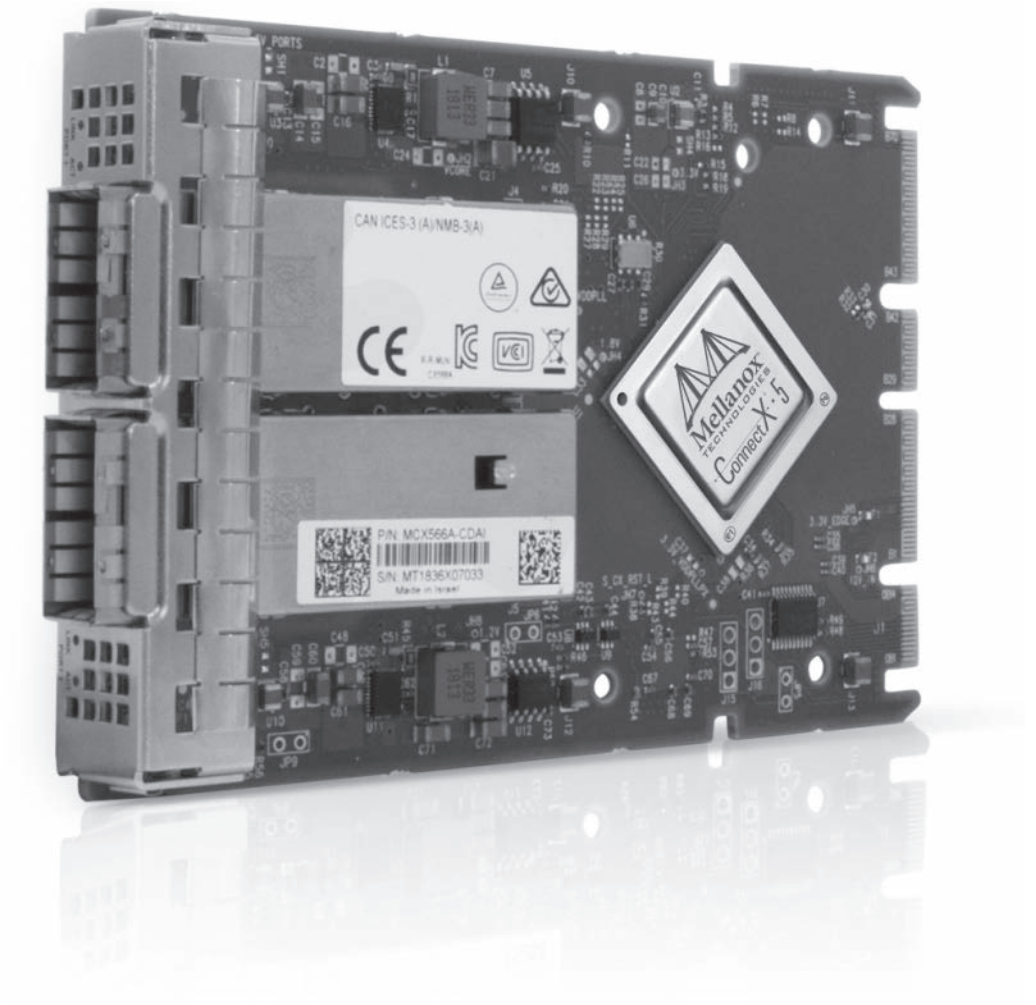



**FIGURE 8-6** Mellanox ConnectX-5 in OCP 3.0 Form Factor (Courtesy of the Open Compute Project Foundation)

## 8.2.4 Lan On Motherboard

It is pretty standard for some class of servers to have a LAN On Motherboard (LOM); that is, to solder on the motherboard the same ASIC used on a NIC card, sometimes also in conjunction with some other chips, such as NIC memories.

LOMs are very important on cost-sensitive servers where the cost of the slot, the connector, and the NIC card are difficult to justify.

Figure 8-7 shows a LOM on a Cisco UCS C220 M5 Server, with two ports at 10 Gbps.

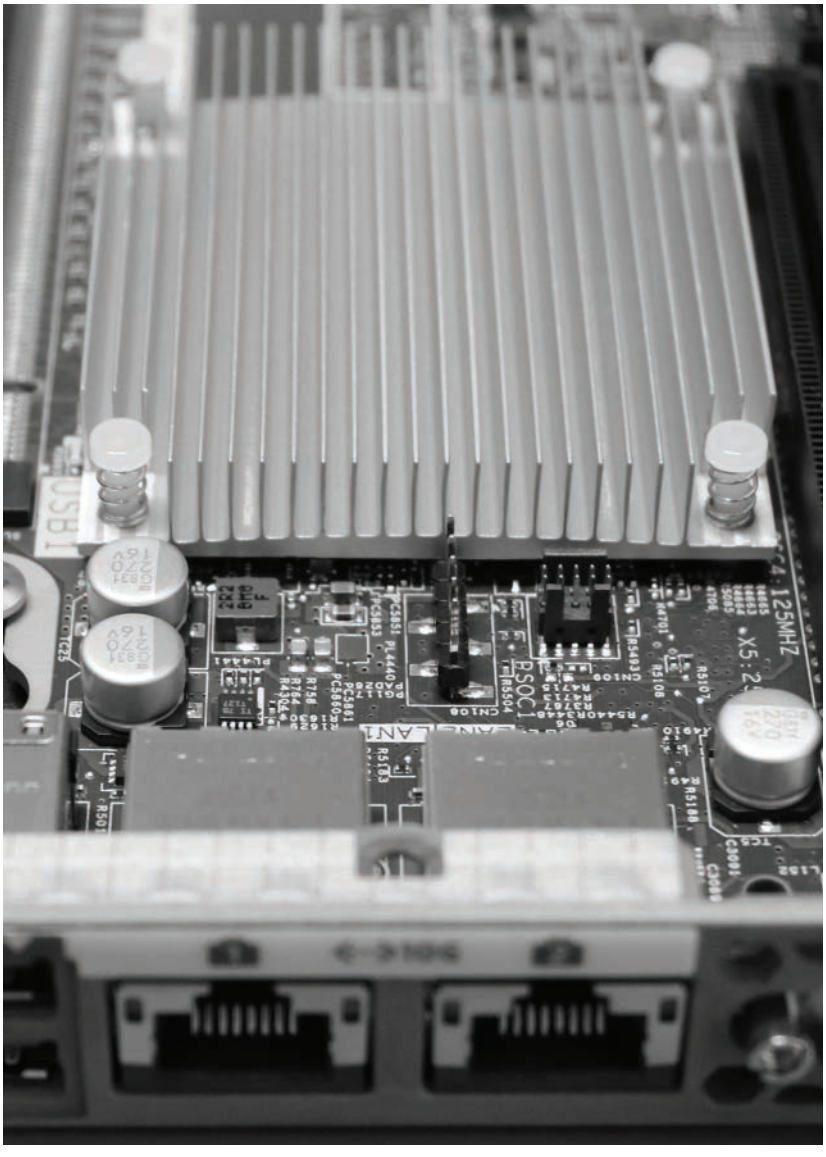



**FIGURE 8-7** LOM on a Cisco UCS

## 8.3 Looking at the NIC Evolution

If we look at the evolution of NICs until 2005, the focus has been to increase the speed of ports, increase the number of ports, support new faster buses (PCI, PCI-X, PCIe), and adapt to new form factors, including reducing power consumption to fit in server expansion slots.

Until that time, NIC architecture had remained pretty constant, mostly based on a simple state machine in charge of processing packets in order: one packet in, one packet out.

Starting around 2005, NICs have gained new functionalities, such as:

- **Multiple queues:** To add Quality of Service (QoS) features, the NIC introduced multiple queues to process different kinds of packets with different priorities. With queues also came a scheduler based on an algorithm such as Deficit Weighted Round Robin (DWRR). There has been a continuous trend towards increasing the number of queues to better support multicore CPUs and application-based queueing. The NIC can assign different network flows to different queues and different queues to different CPUs and cores in the system, thus enabling the system to process the network traffic in parallel, achieving greater network throughput.
- **Traffic shaping:** Traffic shaping is a feature related to QoS to shape the outgoing traffic according to a predefined profile.
- **Multicast handling:** With the increased utilization of multicast applications, in particular in the financial sector, the NIC introduced the capability to support multiple multicast groups and to use protocols such as IGMP to join and leave such multicast groups.
- **Support for TCP/UDP/IP stateless offloads:** These should not be confused with TCP Termination, Stateful Offload, or Proxy that are much more complex features. The three common stateless offloads are:
  - *Checksum offload,* a technique that delegates the computation of checksums to the NIC.
  - *Large send offload (LSO),* a technique for reducing CPU utilization while at the same time increasing egress network throughput. It works by passing a large buffer to the NIC, which splits this buffer into separate packets. This technique is also called TCP segmentation offload (TSO) when applied to TCP, or generic segmentation offload (GSO). For example, a CPU may pass a 64KB buffer and a TCP protocol header template to the NIC. The NIC segments the payload, for example, into 45 segments of 1460 bytes each, and appends the proper Ethernet/IP/TCP header to each segment before sending them out.
  - *Large receive offload (LRO),* the companion technique of LSO on the receiving side. It aggregates multiple incoming packets from a single flow into a single buffer that is passed higher up the networking stack to reduce the number of packets that have to be processed by the OS. Linux implementations generally use LRO in conjunction with the New API (NAPI), an interface that provides interrupt mitigation techniques for networking devices in the Linux kernel. This feature is also called Receive Side Coalescing (RSC).
- **Receive Side Scaling (RSS):** In multicore processors, before this feature, all the packets/interrupts from a NIC went to the same CPU/core, and this created a bottleneck on that CPU/core. RSS enables the spreading of the packets received among all the CPUs/cores in the system. This spreading is typically done using a hashing function on the five-tuples (IP addresses, protocol type, TCP/UDP ports) so that all the packets belonging to the same flow go to the same CPU/core.

- **Message Signaled Interrupts-Extended (MSI-X):** Before MSI-X, the NIC was asserting interrupts to the CPU using dedicated interrupt lines, requiring separate pins on the NIC, on the processors, and all the connectors. With MSI-X, interrupts are signaled using in-band messages. MSI-X is the sole mechanism to signal interrupts on PCIe. MSI-X was introduced in PCIe 3.0 and supports up to 2048 interrupts. MSI-X allows the spreading of interrupts to multiple CPUs and cores.
- **Interrupt moderation:** The idea behind interrupt moderation is that in a high traffic environment, it is counterproductive to interrupt the CPU per each packet. A single interrupt can serve multiple packets. With interrupt moderation, the NIC hardware will not generate an interrupt immediately after it receives a packet—it will wait for more packets to arrive or until a timeout expires.
- **VLAN tagging support:** Virtual LANs (VLANs) are a well-established reality in modern networks. IEEE standardized them as IEEE 802.1Q and, for this reason, the tag in the Ethernet frame is also called "dot 1Q Tag." Modern NICs support one or multiple levels of VLAN tags.
- **Overlay support:** This is an extension of the VLAN tagging support, and it refers to the capability to insert an overlay header in the frame to create a tunnel. For example, in VXLAN the NIC may act as a VTEP (see section 2.3.4).
- **Packet filtering and replication:** To better support virtualization, most NICs added packet filtering and replication. Examples of filtering and replication modes are:
  - By VLAN tag
  - By Ethernet unicast, multicast, or broadcast addresses
  - By replicating a packet to multiple VMs
  - By various mirroring modes
- **Support for DCB:** The term *data center bridging* (DCB) is used by IEEE to collectively indicate Priority-based Flow Control (PFC), Data Center Bridging eXchange (DCBX), and bandwidth management (Enhanced Transmission Selection [ETS]). In the past, these techniques have also been called CEE (Converged Enhanced Ethernet) and Data Center Ethernet (DCE). The most important and controversial of the three is clearly PFC, also known as Per Priority Pause (PPP). With PFC, a physical link can be partitioned into multiple logical connections (by extending the IEEE 802.1Q priority concept), and each priority can be configured to have either lossless or lossy behavior (lossy being the Ethernet default).
- **Support for unified/converged fabric:** The primary motivation for DCB is to support a unified and converged fabric. This term indicates using the same Ethernet infrastructure not only for "native" network traffic (mainly the IP protocol), but also for storage and RDMA traffic. Storage and RDMA traffic don't tolerate packet loss as TCP/IP does. The idea behind DCB was to create logical links that do not lose packets for this kind of traffic. See section 6.1.8 for more details.

- **Support for time synchronization:** Server time synchronization is becoming more important every day to track and correlate what happens on the networks. Initially, the NTP protocol run on the server CPU was precise enough and easy to implement, but the requirement to reach microsecond precision and even 100 nanosecond precision makes NTP inadequate. The next step is to use Precision Time Protocol (PTP), which is part of IEEE 1588. PTP is a protocol used to synchronize clocks throughout a local area network where it achieves clock accuracy in the submicrosecond range. For this to be true, the NIC needs to provide support for it.
- **Support for SR-IOV:** SR-IOV (Single Root I/O Virtualization), discussed in detail in the next section, is based on the idea to integrate a network switch into the NIC to do the functions that are generally done in software by a virtual switch; for example, in a hypervisor.

## 8.4 Using Single Root Input/Output Virtualization

Single Root Input/Output Virtualization (SR-IOV) is a standard developed by PCI-SIG and used in conjunction with virtual machines and hypervisors [4]. It has been around for more than ten years, and it is gaining ground in the cloud offering. SR-IOV allows different virtual machines running on the same hypervisor/server (single root complex) to share a single piece of PCI Express hardware without requiring the hypervisor to implement a software network switch.

SR-IOV defines physical functions (PFs) and virtual functions (VFs). PFs are full-featured PCIe functions, and they are discovered, managed, and manipulated like any other PCIe device. VFs are simpler, and they only provide input/output functionality. An SR-IOV device can have a few PFs (for example, sixteen) and multiple VFs per PF (for instance, 256 VFs/PF). A total limit on the number of VFs is also present, usually 1024.

Figure 8-8 shows the SR-IOV architecture and the relation between the physical NIC, one PF associated with the hypervisor network driver, and two VFs associated with two virtual NICs inside two VMs. This is a hardware alternative to the hypervisor virtual switch: In SR-IOV the switch is in the NIC, in contrast to being in the hypervisor.

The SR-IOV architecture specifies a simple layer 2 switch, and as Chapter 2, “Network Design,” discussed, this is easy to implement because it requires only a binary matching engine. Nothing prevents implementing a more sophisticated layer 3 or layer 4 forwarding in the SR-IOV switch. From a data plane perspective, an approach such as LPM forwarding or flow table forwarding can be used. From a control and management plane perspective, the switch can be programmed; for example; using a RESTful API or using an OpenFlow controller (for example, the Faucet controller [5]). Another alternative is to use OVSDB to program the switch inside SR-IOV (see section 4.3.1).

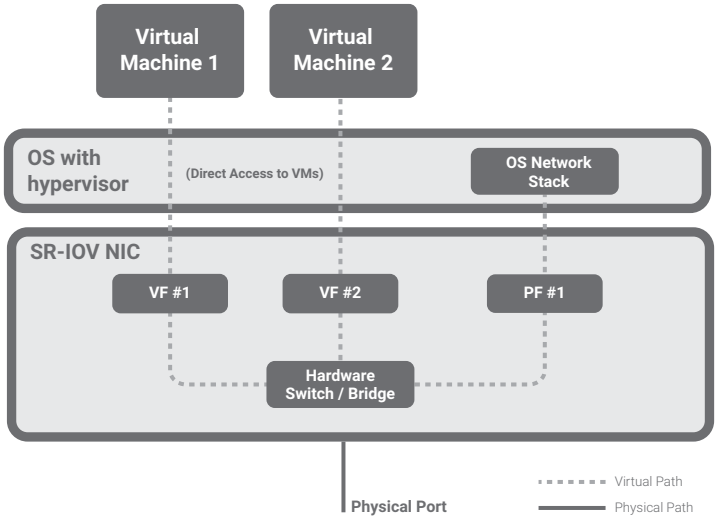



**FIGURE 8-8** SR-IOV Architecture

The control and management planes may run on the server CPUs, or they can run on a core embedded in the NIC. The same is true for the first packet processing, in the case of the flow table approach.

Again, the level of server CPU offload obtained will depend on these choices.

## 8.5 Using Virtual I/O

Virtual I/O (VirtIO) [6], [7] is an OASIS project [8] to simplify virtual devices, making them more extensible, recognizable, and supported across multiple operating systems. It was introduced in section 3.2.4. VirtIO PCI devices are very similar to physical PCI devices, and, therefore, they can be used both by physical NICs and virtual NICs. Standard drivers for them have been upstreamed in the major operating systems. Two popular VirtIO devices are virtio-blk (for block storage) and virtio-net (for network communications). The other three are PCI emulation, a balloon driver (for dynamically managing guest memory usage), and a console driver. Figure 8-9 shows the VirtIO architecture.

At the top are the front-end drivers implemented, for example, in the VM operating system. At the bottom there are the back-end drivers implemented, for example, in the hypervisor. Each front-end driver has a corresponding back-end driver. Two additional layers in between implement queueing among other services.

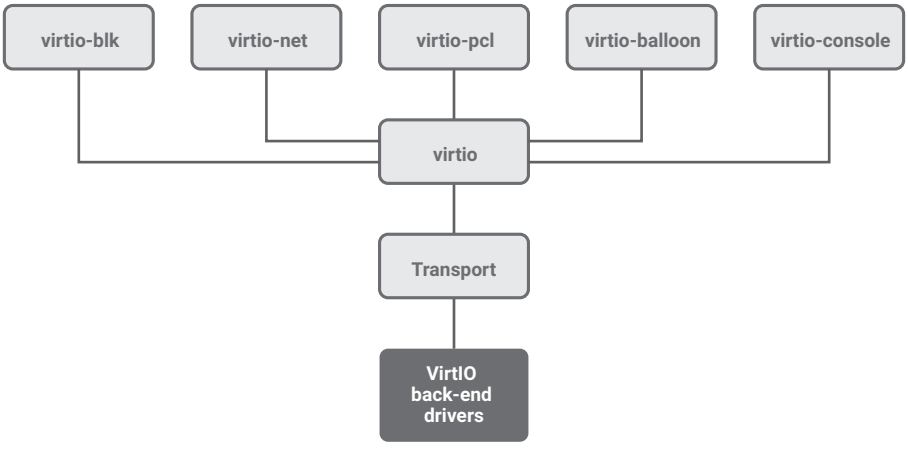



**FIGURE 8-9** VirtIO Architecture

When the front-end drivers are used in conjunction with a virtualization environment, they are aware that they are running in a virtual environment and cooperate with the hypervisor to improve performance.

## 8.6 Defining "SmartNIC"

Up to now, we have discussed the evolution of classical NICs. Around 2016, the term *SmartNIC* was introduced, although there is no clear and broadly accepted definition of that term. One of the best and simpler definitions is "a NIC that offloads processing tasks that the system CPU would normally handle."

To achieve this goal, a SmartNIC typically includes some of these features:

- **Processors:** These are generally in the form of ARM cores and usually run control and management protocols.
- **L2/L3 switching, regularly using SR-IOV:** The goal is to offload switching in hardware, but the level of offload obtained varies significantly with implementations.
- **Support for SDN flow tables:** Similarly to the previous point, flow tables can be used to offload switching from the host to the NIC.
- **Network overlays (for example, IP in IP, VXLANs):** These are particularly relevant in cloud environments, where almost all the packets need to transit through an overlay network.
- **RDMA:** This is relevant for machine learning, high-performance computing (HPC), and other tasks that require distributing large datasets, and for remote NVMe storage.
- **Storage protocols with I/O consolidation:** The NIC exposes NVMe interfaces to the host and supports directly attached disks or remote (virtual) disks.

Even if these features should be independent of the speed of the links, due to their recent introduction on the market, SmartNICs typically are 100 Gbps devices with a few 100 GE ports, also configurable as 25GE and 50GE.

Examples of 100 Gbps SmartNIC are:

- The Broadcom BCM58800 - Stingray SmartNIC and Storage Controller IC that includes eight ARMv8 Cortex-A72 CPUs at 3.0 GHz and three memory channels DDR4-2400 [9]
- The Mellanox BlueField Multicore System-on-Chip (SoC) that includes 16 ARMv8 A72 cores and two DDR4 memory channels [10]

## 8.7 Summary

In this chapter, we have discussed the evolution of the NIC from a simple packet-passing device to domain-specific hardware that can implement features of a distributed-services platform. We have explained that one of the main issues is power consumption, which is limited by thermal dissipation.

In the next chapter, we compare the advantages and disadvantages of implementing features in the NIC or another part of the network.

## 8.8 Bibliography

[1] Ravi Budruk, "PCI Express Basics," PCI-SIG presentation, 2007-08-21, archived on Wikipedia, https://en.wikipedia.org/wiki/PCI_Express

[2] Zale Schoenborn, "Board Design Guidelines for PCI Express Architecture," 2004, PCI-SIG, pp. 19–21, archived on Wikipedia, https://en.wikipedia.org/wiki/PCI_Express

[3] Open Compute Project, http://www.opencompute.org

[4] Single Root I/O Virtualization and Sharing, PCI-SIG.

[5] "Faucet: Open source SDN Controller for production networks," https://faucet.nz

[6] VirtIO, https://wiki.libvirt.org/page/Virtio

[7] libvirt, libvirt. https://libvirt.org

[8] Oasis, Open Standards, Open Source, https://www.oasis-open.org

[9] Broadcom, "Stingray SmartNIC Adapters and IC," https://www.broadcom.com/products/ethernet-connectivity/smartnic

[10] Mellanox, "BlueField SmartNIC for Ethernet" https://www.mellanox.com/related-docs/prod_adapter_cards/PB_BlueField_Smart_NIC_ETH.pdf