IIR

Internet
Infrastructure
Review

Dec.2023
Vol. 60

Periodic Observation Report

Broadband Traffic Report
Traffic in a Stable Uptrend Post-COVID

Focused Research(1)

A Review of Research in System Software Communications Since 2010

Focused Research(2)

Evolution of the IIJ Cloud
—Commemorating 30 Years

IIJ
Internet Initiative Japan

Internet Infrastructure Review

December 2023 Vol.60

© Internet Initiative Japan Inc.

Executive Summary

This past July, Japan’s ruling party, the Liberal Democratic Party (LDP), announced that it would begin a full review of the case for and against selling off shares in NTT held by the government. Japan’s NTT Act currently requires that the government own at least a third of NTT shares, and the LDP is looking to make recommendations on the future status of that Act and on the issue of enhancing the international competitiveness of Japan’s telecommunications industry overall as early as this fall.

Japan’s Ministry of Internal Affairs and Communications has also asked the Information and Communications Council under its auspices to conduct a study into how telecommunications policy should be structured in response to changes in the market environment, and to report its findings around next summer. The Council set up a special committee on telecommunications policy in response, through which it has put out a broad call for proposals and opinions regarding the vision Japan should pursue for its information and communications infrastructure circa 2030 and the basic policy directions.

Telecommunications used to be a state-owned enterprise in Japan, run by NTT. The privatization of NTT and resulting influx of competition came in 1985. Since then, the environment surrounding information and communications in Japan has seen dramatic change, including with respect to the way in which telecommunications are used, the services and benefits provided, the position and importance of telecommunications in society, and the players that populate the commercial landscape. The Internet has, without a doubt, been an influencing factor in all of this.

NTT naturally continues to be an important player in Japan’s information and communications infrastructure, but beyond this, I also look forward to people engaging in high-level discussion and debate ahead about the nature of information and communications not only in Japan but around the world.

The IIR introduces the wide range of technology that IIJ researches and develops, comprising periodic observation reports that provide an outline of various data IIJ obtains through the daily operation of services, as well as focused research examining specific areas of technology.

The periodic observation report in Chapter 1 is our broadband traffic report for the year, providing our analysis of IIJ’s fixed broadband and mobile traffic. Although we have observed no notable changes since the COVID-19 pandemic sparked significant traffic growth from 2020, the figures do clearly indicate that traffic volumes and usage by port are steadily changing.

Chapter 2 presents a focused research report that dives into efforts to speed up communications processing on systems software, titled “A Review of Research in Systems Software Communications Since 2010”. As the processing power of hardware improves, the efficiency of systems software that controls that hardware, particularly the communications-related processing it performs, is becoming increasingly important. The report starts with an overview of how systems software processes communications data, then surveys past research aimed at making this more efficient, and closes out with a look at IIJ Research Laboratory’s own efforts in this area.

Our focused research report in Chapter 3 continues our series commemorating IIJ’s 30-year history. Previous installments covered the IIJ backbone and DNS, and here we take a journey through the evolution of IIJ’s cloud services. Since its founding, IIJ has been operating service hosts that provide a whole range of services to accompany its Internet connectivity offerings. And even before terms like “the cloud” and “IaaS” entered the common vernacular, IIJ had been providing computing and storage resources as services. The infrastructure that encompasses the servers, storage systems, and networks used by these services has continued to evolve with the times, and our report here goes over the changes that have occurred.

Through activities such as these, IIJ strives to improve and develop its services on a daily basis while maintaining the stability of the Internet. We will continue to provide a variety of services and solutions that our customers can take full advantage of as infrastructure for their corporate activities.

Junichi Shimagami

Mr. Shimagami is a Senior Executive Officer and the CTO of IIJ. His interest in the Internet led to him joining IIJ in September 1996. After engaging in the design and construction of the A-Bone Asia region network spearheaded by IIJ, as well as IIJ’s backbone network, he was put in charge of IIJ network services. Since 2015, he has been responsible for network, cloud, and security technology across the board as CTO. In April 2017, he became chairman of the Telecom Services Association of Japan’s MVNO Council, stepping down from that post in May 2023. In June 2021, he also became a vice-chairman of the association.

© Internet Initiative Japan Inc.

Broadband Traffic Report
Traffic in a Stable Uptrend Post-COVID

1.1 Overview

In this report, we analyze traffic over the broadband access services operated by IIJ and present the results each year[*1][*2][*3][*4][*5]. Here, we again report on changes in traffic trends over the past year, based on daily user traffic and usage by port. Traffic has continued to grow steadily since the COVID-19 pandemic fell into the rearview mirror, and at present we see no noticeable changes in that overall trend.

Figure 1 plots the overall average monthly traffic trends for IIJ's fixed broadband services and mobile services. IN/OUT indicates the direction from the ISP perspective. IN represents uploads from users, and OUT represents user downloads. Because we cannot disclose specific traffic numbers, we have normalized the data, setting the OUT observations for January 2020, just before the pandemic, for both services to 1.

Over the past year, broadband IN traffic increased 11% and broadband OUT traffic increased 18%. The corresponding year-earlier figures were 13% and 17%.

The broadband figures include IPv6 IPoE traffic. IPv6 traffic on IIJ's broadband services comprises both IPoE and PPPoE traffic. As of June 2023, IPoE accounted for over 40% of all traffic, at 42% of IN and 44% of OUT broadband traffic overall, year-on-year increases of 3 percentage points each. With PPPoE congestion having become quite noticeable amid COVID-19, users are increasingly shifting to IPoE, and use of IPoE thus continues to rise.

Mobile services traffic has been in an uptrend since 2021. Over the past year, mobile IN traffic increased 27% and mobile OUT traffic increased 31%.

We now look at broadband traffic by time of day on weekdays over the past year. Figure 2 plots hourly average traffic volume for Monday–Friday for four one-week blocks selected at intervals of roughly four months since May 2022. Weekday daytime traffic volumes have increased during school holiday periods in recent years, so we selected school weeks. Traffic volume here is the sum of PPPoE and IPoE. The dotted lines in the lower part of the plot

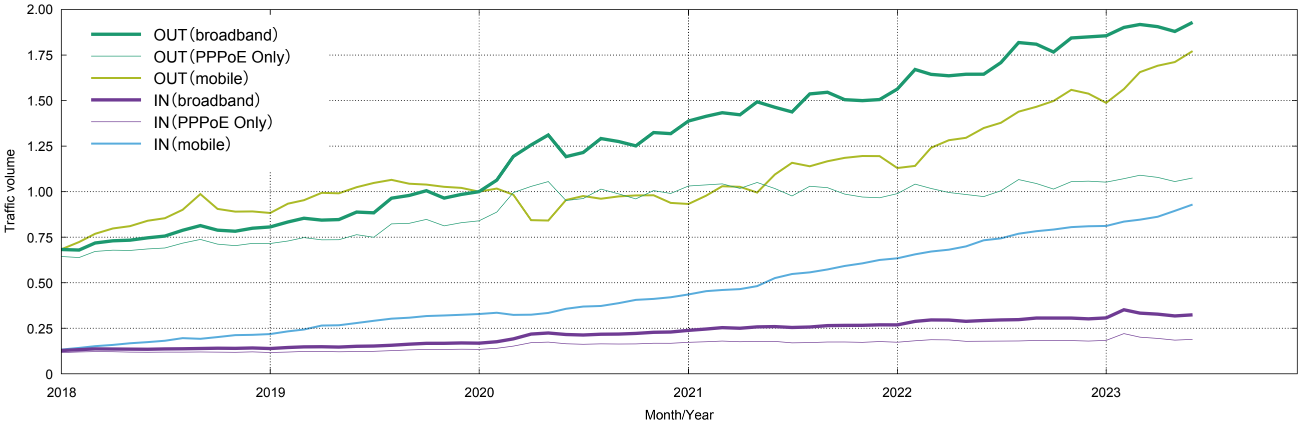

Figure 1: Monthly Broadband and Mobile Traffic

[*1] Kenjiro Cho. Broadband Traffic Report: Broadband Traffic Report: COVID's 3rd Year Brings Lull in Traffic. Vol. 56. pp4-11. September 2022.
[*2] Kenjiro Cho. Broadband Traffic Report: Broadband Traffic Report: COVID-19's Impact in its 2nd Year. Vol. 52. pp4-11. September 2021.
[*3] Kenjiro Cho. Broadband Traffic Report: The Impact of COVID-19. Vol. 48. pp4-9. September 2020.
[*4] Kenjiro Cho. Broadband Traffic Report: Moderate Growth in Traffic Volume Ongoing. Vol. 44. pp4-9. September 2019.
[*5] Kenjiro Cho. Broadband Traffic Report: Download Growth Slows for a Second Year Running. Vol. 40. pp4-9. September 2018.

© Internet Initiative Japan Inc.

represent uploads for each week, but focusing again on download volumes in this edition, we see that traffic volumes were up across all times of the day. The size of the increase during the middle of the day and during the nighttime peak are around the same in absolute terms, so the proportional increase is higher for daytime traffic.

1.2 About the Data

As with previous reports, for broadband traffic, our analysis uses data sampled using Sampled NetFlow from the routers that accommodate the fiber-optic and DSL broadband customers of our personal and enterprise broadband access services. For mobile traffic, we use access gateway billing information to determine usage volumes for personal and enterprise mobile services, and we use Sampled NetFlow data from the routers used to accommodate these services to determine the ports used.

Because traffic trends differ between weekdays and weekends, we analyze traffic in one-week chunks. In this report, we look at data for the week of May 29 – June 4, 2023, and compare those data with data for the week of May 30 – June 5, 2022, which we analyzed in the previous edition of this report.

Results are aggregated by subscription for broadband traffic, and by phone number for mobile traffic as some subscriptions cover multiple phone numbers. The usage volume for each broadband user was obtained by matching the IP addresses assigned to users with the IP addresses observed. We gathered statistical information by sampling packets using NetFlow. The sampling rate was set to around 1/8,192, taking into account router performance and load. We estimated overall usage volumes by multiplying observed volumes by the reciprocal of the sampling rate. Note that IPoE traffic is not included in the analysis of traffic by port, as detailed data is not available because we use Internet Multifeed Co.'s transix service for IPoE.

1.3 Users' Daily Usage

First, we examine daily usage volumes for broadband and mobile users from several angles. Daily usage indicates the average daily usage calculated from a week's worth of data for each user.

Since our 2019 report, we have used daily usage data only on services provided to individuals. The distribution is heavily distorted if we include enterprise services, where usage patterns are highly varied. So to form a picture of overall usage trends, we determined that using only the personal user data would yield more generally applicable, easily interpretable conclusions. Note that the analysis of usage by port in the next section does include enterprise data because of the difficulty of distinguishing between individual and enterprise usage. Note also that we have included IPoE user data in the broadband figures since 2021. In the previous edition of this report, we showed PPPoE and IPoE separately, but starting with this edition, we roll both sets of figures into a single broadband data set[*6].

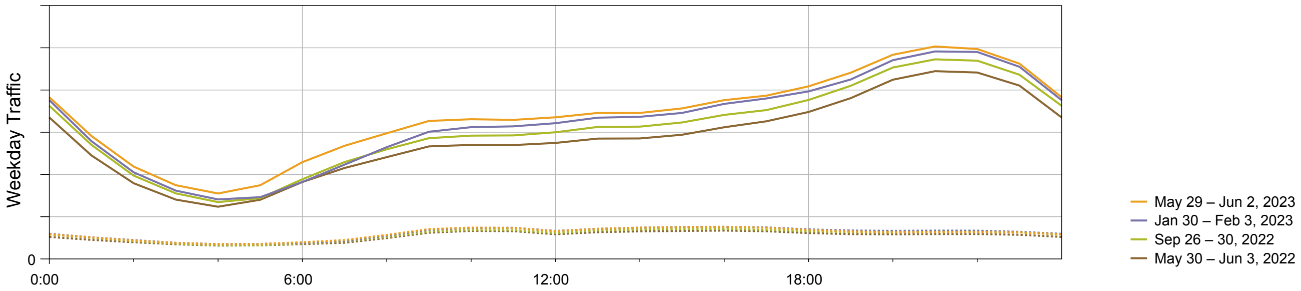

Figure 2: Hourly Average Broadband Traffic on Weekdays in the Past Year

[*6] The PPPoE and IPoE usage figures of users who use both protocols are treated as coming from separate users.

Figures 3 and 4 show the average daily usage distributions (probability density functions) for broadband and mobile users. Each compares data for 2022 and 2023 split into IN (upload) and OUT (download), with user traffic volume plotted along the X-axis and user frequency along the Y-axis. The X-axis shows volumes between 10KB (10^4) and 100GB (10^{11}) using a logarithmic scale. Most users fall within the 100GB (10^{11}) range, with a few exceptions.

The IN and OUT traffic distributions in the figures are close to a log-normal distribution, which looks like a normal distribution on a semi-log plot. A linear plot would show a long-tailed distribution, with the peak close to the left and a slow gradual decrease toward the right. The OUT distribution is further to the right than the IN distribution, indicating that download volume is more than an order of magnitude larger than upload volume.

First, we look at the broadband distributions in Figure 3. Comparing 2022 and 2023, both the IN and OUT distributions have moved just slightly to the right, indicating that overall traffic has increased. The peaks of the mobile distributions in Figure 4 have also moved a little to the right since last year, again indicating that overall traffic has increased. Mobile usage volumes are significantly lower than for broadband, and limits on mobile data usage mean that heavy users, which fall on the right-hand side of the distribution, account for only a small proportion of the total. There are also no extremely heavy users. The variability in each user's daily usage volume is higher for mobile than for broadband owing to there being users who only use mobile data when out of the home/office as well as limits on mobile data.

Table 1 shows trends in the mean and median daily traffic values for broadband users as well as the mode (the most frequent value, which represents the peak of the distribution). When the peak is slightly off the center of the distribution, the distribution is adjusted to bring the mode

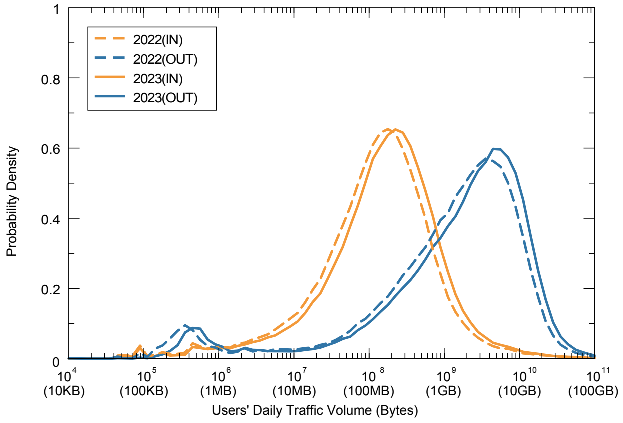

Figure 3: Daily Broadband User Traffic Volume Distribution Comparison of 2022 and 2023

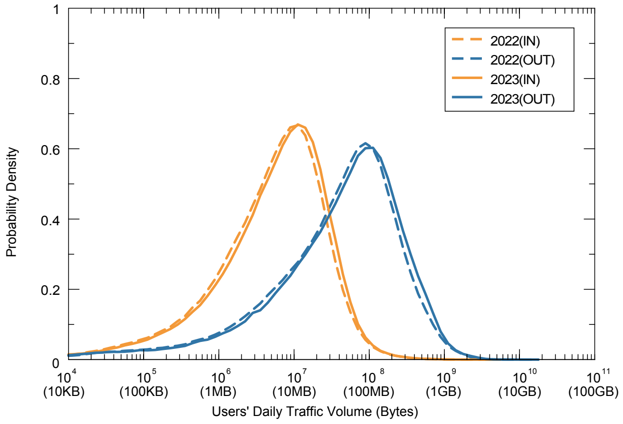

Figure 4: Daily Mobile User Traffic Volume Distribution Comparison of 2022 and 2023

Table 1: Trends in Mean and Mode of Broadband Users' Daily Traffic Volume

	IN(MB/day)			OUT(MB/day)		
Year	Mean	Median	Mode	Mean	Median	Mode
2007	436	5	5	718	59	56
2008	490	6	6	807	75	79
2009	561	6	6	973	91	100
2010	442	7	7	878	111	126
2011	398	9	9	931	144	200
2012	364	11	13	945	176	251
2013	320	13	16	928	208	355
2014	348	21	28	1124	311	501
2015	351	32	45	1399	443	708
2016	361	48	63	1808	726	1000
2017	391	63	79	2285	900	1259
2018	428	66	79	2664	1083	1585
2019	479	75	89	2986	1187	1995
2020	609	122	158	3810	1638	3162
2021	714	143	200	4432	2004	3981
2022	727	142	178	4610	2010	3981
2023	804	166	224	5456	2369	5012

toward the center. Comparing 2022 and 2023, the IN mode rose from 178MB to 224MB and the OUT mode rose from 3,981MB to 5,012MB, translating into growth factors of 1.26 for IN and 1.26 for OUT. Meanwhile, because the means are influenced by heavy users (on the right-hand side of the distribution), they are significantly higher than the corresponding modes, with the IN mean at 804MB and the OUT mean at 5,456MB in 2023. The 2022 means were 727MB and 4,610MB, respectively. As mentioned, up to 2020 the data covered only PPPoE users, and since 2021 the data have covered both PPPoE and IPoE users.

Table 2 shows the mobile traffic metrics. In 2023, the IN mode was 11MB and the OUT mode was 100MB, while the means were IN 14MB and OUT 129MB. The 2022 modes were IN 10MB and OUT 89MB, and the means were IN 13MB and OUT 114MB.

Table 2: Trends in Mean and Mode of Mobile Users’ Daily Traffic Volume

	IN(MB/day)			OUT(MB/day)		
Year	Mean	Median	Mode	Mean	Median	Mode
2015	6.2	3.2	4.5	49.2	23.5	44.7
2016	7.6	4.1	7.1	66.5	32.7	63.1
2017	9.3	4.9	7.9	79.9	41.2	79.4
2018	10.5	5.4	8.9	83.8	44.3	79.4
2019	11.2	5.9	8.9	84.9	46.4	79.4
2020	10.4	4.5	7.1	79.4	35.1	63.1
2021	9.9	4.7	7.9	85.9	37.9	70.8
2022	12.8	6.0	10.0	113.7	49.2	89.1
2023	14.1	6.8	11.2	129.2	56.0	100.0

Figures 5 and 6 plot per-user IN/OUT usage volumes for random samples of 5,000 users. The X-axis shows OUT (download volume) and the Y-axis shows IN (upload volume), with both using a logarithmic scale. Users with identical IN/OUT values fall on the diagonal.

The cluster spread out below and parallel to the diagonal in each of these plots represents typical users with download volumes an order of magnitude higher than upload volumes. Variability between users in terms of usage levels and IN/OUT ratios is wide, indicating that there is a diverse range of usage styles. For mobile traffic, the pattern of OUT being an order of magnitude larger also applies, but usage volumes are much lower than for broadband. For both broadband and mobile, there is almost no difference between these plots and those for 2022.

Traffic is heavily skewed across users, such that a small proportion of users accounts for the majority of overall traffic volume. For example, the top 10% of broadband users account for 49% of total OUT and 76% of total IN traffic, while the top 1% of users account for 16% of OUT and 49% of IN traffic. On mobile, the top 10% of users account for 49% of total OUT and 47% of total IN traffic, while the top 1% of users account for 15% of OUT and 15% of IN traffic.

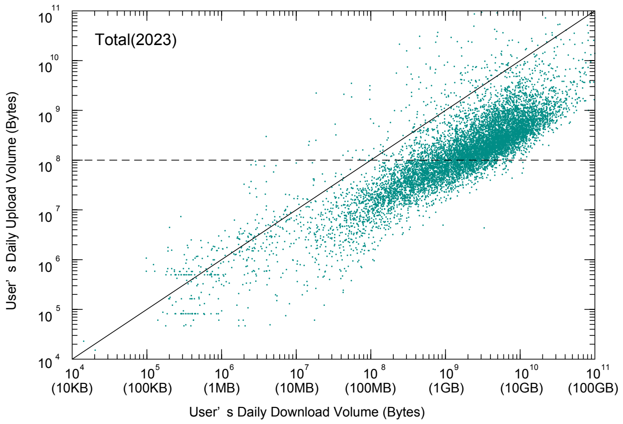

Figure 5: IN/OUT Usage for Each Broadband User

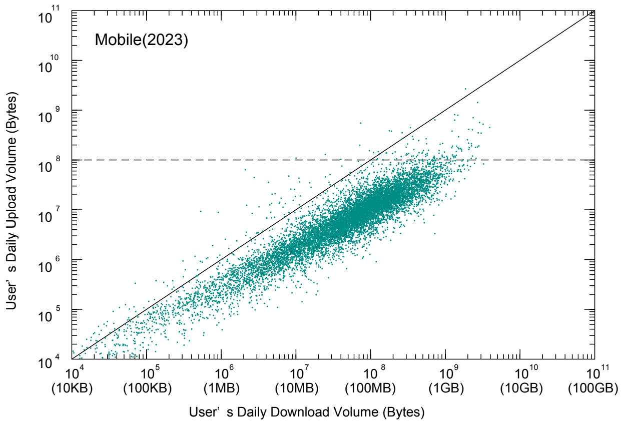

Figure 6: IN/OUT Usage for Each Mobile User

1.4 Usage by Port

Next, we look at a breakdown of traffic and examine usage levels by port. Recently, it has become difficult to identify applications by port number. Many P2P applications use dynamic ports on both ends, and a large number of client/server applications use HTTP ports like port 80 to avoid firewalls. Hence, generally speaking, when both parties are using a dynamic port numbered 1024 or higher, the traffic is likely to be from a P2P application, and when one of the parties is using a well-known port lower than 1024, the traffic is likely to be from a client/server application. In light of this, we take the lower of the source and destination port numbers when breaking down TCP and UDP usage volumes by port.

Table 3 shows the percentage breakdown of broadband users' usage by port over the past five years. In 2023, 71% of all traffic was over TCP connections, down 1 point from 2022. The proportion of traffic over port 443 (HTTPS) was 57%, a 1-point increase from last year. The proportion of traffic over port 80 (HTTP) fell from 9% to 7%. The figure for UDP port 443, which is used by the QUIC protocol, was up 2 points to 18%.

TCP dynamic port traffic fell ever so slightly to below 6%. Individual dynamic port numbers account for only a tiny portion, with the most commonly used port 31000 only making up 1.1%.

Table 4 shows the percentage breakdown by port for mobile users. The figures are close to those for broadband on the whole. This is possibly because apps similar to those for PC platforms are now also used on smartphones, and because the proportion of broadband usage on smartphones is rising.

The broadband port data only include PPPoE, not IPoE, and so do not necessarily reflect the trend in fixed broadband overall. Comparing IPv4 and IPv6 on mobile, port 443 accounts for a higher proportion of both TCP and UDP usage on IPv6, and there is probably a similar trend in the case of IPoE.

Table 3: Broadband Users' Usage by Port

year	2019	2020	2021	2022	2023
protocol port	(%)	(%)	(%)	(%)	(%)
TCP	**81.2**	**77.2**	**71.9**	**71.6**	**70.5**
(< 1024)	73.3	70.5	65.8	65.4	64.8
443(https)	51.9	52.4	53.5	55.7	56.9
80(http)	20.4	17.2	11.6	8.9	7.2
183	0.0	0.0	0.1	0.2	0.2
993(imaps)	0.3	0.2	0.1	0.1	0.1
22(ssh)	0.2	0.2	0.2	0.1	0.1
(>= 1024)	7.9	6.7	6.1	6.2	5.7
31000	0.2	0.4	0.6	0.9	1.1
8080	0.5	0.4	0.4	0.3	0.4
1935(rtmp)	0.3	0.4	0.2	0.2	0.2
UDP	**14.1**	**19.4**	**24.5**	**24.3**	**25.4**
443(https)	7.8	10.5	15.9	16.3	18.2
4500(nat-t)	0.3	0.6	0.8	0.8	1.0
8801	0.0	1.1	0.9	0.6	0.4
ESP	**4.4**	**3.2**	**3.3**	**3.8**	**3.8**
GRE	**0.1**	**0.1**	**0.2**	**0.2**	**0.1**
IP-ENCAP	**0.2**	**0.1**	**0.1**	**0.1**	**0.1**
ICMP	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

Table 4: Mobile Users' Usage by Port

year	2019	2020	2021	2022	2023
protocol port	(%)	(%)	(%)	(%)	(%)
TCP	**76.9**	**75.5**	**70.3**	**71.6**	**71.0**
443(https)	55.6	50.7	44.4	42.3	42.1
80(http)	10.3	7.4	5.0	4.1	3.5
993(imaps)	0.3	0.2	0.2	0.1	0.1
1935(rtmp)	0.1	0.1	0.1	0.1	0.2
UDP	**17.3**	**18.0**	**23.8**	**24.4**	**26.5**
443(https)	8.3	9.3	16.3	17.9	20.9
4500(nat-t)	3.0	1.8	3.7	2.7	2.5
8801	0.0	1.4	0.7	0.3	0.2
51820	0.0	0.0	0.0	0.1	0.2
53(dns)	0.1	0.1	0.2	0.2	0.2
ESP	**5.8**	**6.4**	**5.8**	**3.9**	**2.4**
GRE	**0.0**	**0.1**	**0.1**	**0.0**	**0.0**
ICMP	**0.0**	**0.0**	**0.0**	**0.0**	**0.1**

Figure 7 compares overall broadband traffic for key port categories across the course of the week from which observations were drawn in 2022 and 2023. We break the data into four port buckets: TCP ports 80 and 443, dynamic TCP ports (1024 and up), and UDP port 443. The data are normalized so that peak overall traffic volume on the plot is 1. The overall peak is around 19:00–23:00. When compared, there are no major changes between 2022 and 2023, but traffic on UDP port 443 is up a little, and as previously mentioned in relation to Figure 2, the proportion of traffic accounted for by daytime hours has also increased a bit.

Figure 8 shows the trend for TCP ports 80 and 443 and UDP port 443, which account for the bulk of mobile traffic. As was the case with broadband, mobile traffic on UDP port 443 was up slightly compared with 2022. The lunchtime peak is a little lower compared with 2022 and, accordingly, more spread out. Comparing the plots with those for broadband, usage times evidently differ, with mobile having three separate traffic peaks on weekdays: morning commute, lunch break, and evening.

1.5 Conclusion

With the COVID-19 pandemic behind us, we have finally returned to normal everyday life, and yet the Internet has come to permeate our daily affairs and is now a crucial part of our everyday infrastructure. Video conferencing and remote work are here to stay, and even our children now routinely stream video in the home. And with Japan’s performances in the 2022 FIFA World Cup and the March 2023 World Baseball Classic attracting eyeballs, online broadcasting has also broadened the sports viewership base. The proportion of social media accounted for by video content has also increased markedly relative to a few years ago. Meanwhile, although per-user traffic volume on both broadband and mobile jumped in 2020, the first year of the pandemic, it has since eased to relatively stable growth levels. Overall traffic volume also remains in a solid uptrend. Contributing factors here may include the decline in work-from-home rates since 2021 resulting in people spending less time online, a lack of any truly notable new services or use cases, and milder growth in traffic volumes than growth in video content volumes would otherwise suggest owing to technological advancements such as more efficient video compression.

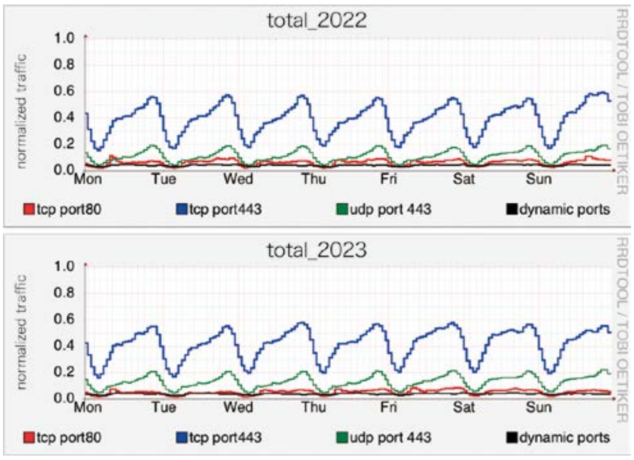

Figure 7: Broadband Users’ Port Usage Over a Week 2022 (top) and 2023 (bottom)

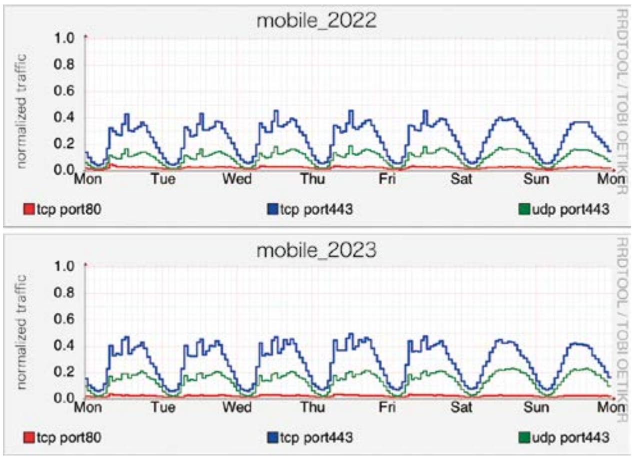

Figure 8: Mobile Users’ Port Usage Over a Week 2022 (top) and 2023 (bottom)

Kenjiro Cho

Research Director, Research Laboratory, IIJ

A Review of Research in System Software Communications Since 2010

2.1 Overview

Network interface cards (NICs) supporting speeds above 10 Gbps became commonplace in the early 2010s and are now widely used in data-center and other applications. With the performance of NICs rising, the efficiency of the system software that controls this hardware, particularly its data communications, has become increasingly important, and the research community has pursued many avenues to improve this performance.

In Section 2.2, I start by looking at the behavior of system software when processing communications. Section 2.3 then summarizes past research aimed at speeding this up. With that background in place, Section 2.4 then looks at IIJ Research Laboratory's efforts in this area in recent years.

I hasten to add, however, that the efforts described in Section 2.4 are still in the research stages and not yet part of IIJ's service infrastructure.

2.2 Main Communications-related Program Behaviors

In Section 2.2.1, I start by walking through communications processing in general-purpose OSes, and then in Section 2.2.2 I discuss communications on virtual machines (VMs) commonly used in data centers.

2.2.1 Communications-related Processing in General-purpose OSes

Let's look at communications processing in a general-purpose OS environment, with reference to Figure 1.

■ Basic system structure

The three main components are (from top to bottom in Figure 1):

(1) Application running in user space
(2) The kernel, which implements the network stack and device drivers
(3) A NIC, which sends and receives packets.

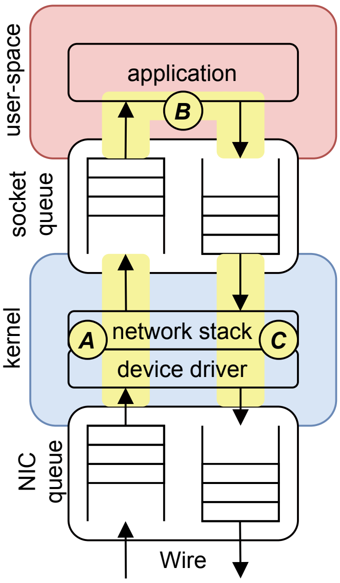

Figure 1: Structure of Communications Software in a General-purpose OS

© Internet Initiative Japan Inc.

■ Typical loop

Programs, called servers, that respond to client requests typically run the following loop: (A) process incoming packets in kernel space, (B) perform application-specific processing in user space, and (C) process outgoing packets in kernel space.

Packet receipt and transmission processing is summarized in Figures 2 and 3 by execution context.

■ A: Incoming packet processing in kernel space

□ STEP 1: Hardware (NIC) notifies software

When a NIC receives a new packet, it issues a hardware interrupt to the CPU to notify the software. This interrupts the program that was running on the CPU and switches to the hardware interrupt handler set up by the kernel in advance. Hardware interrupt handlers are implementation-dependent, but it's fairly common for them to start a kernel thread to process incoming packets.

□ STEP 2: Process incoming packets

The kernel thread started to process the incoming packets in Step 1 reads the incoming packet headers and processes the packets accordingly. For example, if a TCP packet is received, it checks the TCP ACK number for the corresponding connection, and adds the packet to the queue of the socket associated with that connection.

□ STEP 3: Notify user-space process

In Step 2, when data or a new connection is added to a socket queue, if the user-space process/thread associated with that socket is waiting (blocking state) for new input per the select, poll, epoll_wait, or read family of system calls (e.g., read or recvmsg), then the process/thread started (unblocked).

■ B: Program processing in user space

□ STEP 1: Awaiting and detecting input events

Many server programs that run in user space stop execution (remain in a blocking state) when using select, poll, epoll_wait, or read system calls to wait for new input to sockets (file descriptors) they are listening on. If input is received on a socket, this standby (blocking) state is released in Step 3 of A above (incoming packet processing in kernel space). Also, when a system call like select, poll, or epoll_wait unblock execution and return a value, the kernel passes on information about which socket (file descriptor) the input event occurred on.

□ STEP 2: Data passed from kernel to user space

The user-space program issues a read system call to the socket (file descriptor) on which the input event in Step 1 was detected, and then copies the data added to the socket queue in Step 2 of A above (incoming packet processing in kernel space) from the kernel to user space.

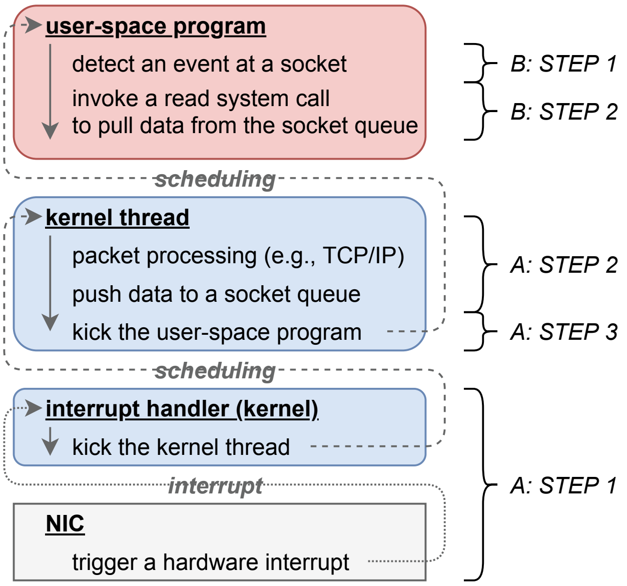
user-space program
detect an event at a socket
invoke a read system call
to pull data from the socket queue
B: STEP 1
B: STEP 2
scheduling
kernel thread
packet processing (e.g., TCP/IP)
push data to a socket queue
kick the user-space program
A: STEP 2
A: STEP 3
scheduling
interrupt handler (kernel)
kick the kernel thread
interrupt
NIC
trigger a hardware interrupt
A: STEP 1

Figure 2: Processing of Incoming Packets on a General-purpose OS —Part and Step Indicated at Right

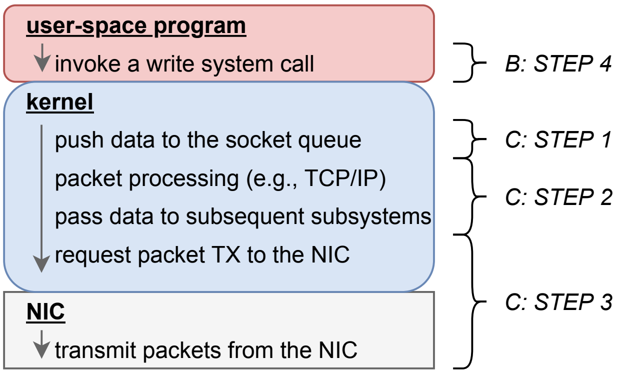
user-space program
invoke a write system call
B: STEP 4
kernel
push data to the socket queue
packet processing (e.g., TCP/IP)
pass data to subsequent subsystems
request packet TX to the NIC
C: STEP 1
C: STEP 2
C: STEP 3
NIC
transmit packets from the NIC

Figure 3: Processing of Outgoing Packets on a General-purpose OS— Part and Step Indicated at Right

□ **STEP 3: Application-specific processing**

The program performs its application-specific processing on the data received in Step 2. For example, a web server would parse the received data, determine the content of the request, and then generate response data.

□ **STEP 4: Tell the kernel to send the data**

The program issues a write system call (e.g., write or sendmsg) to the socket (file descriptor) that tells the kernel to send the data generated in Step 3.

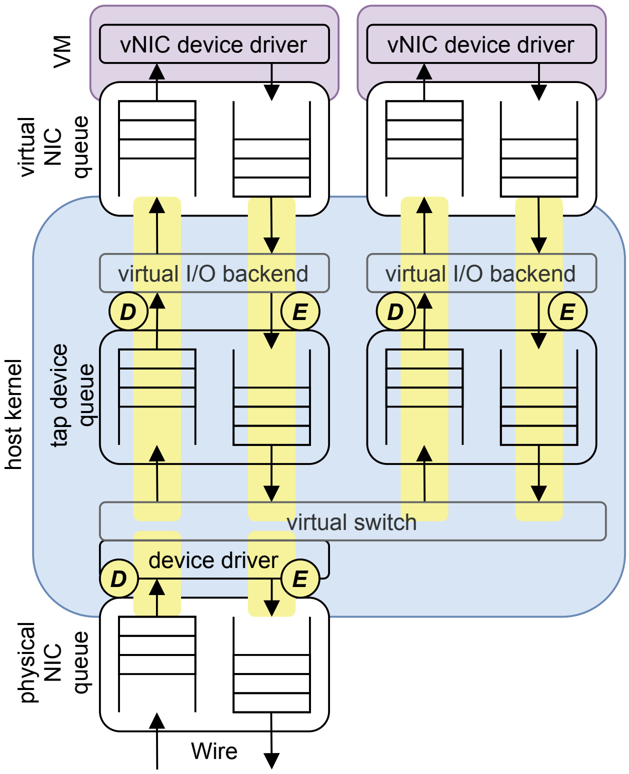

Figure 4: Virtual Machine Communications Mechanism

■ **C: Outgoing packet processing in kernel space**

□ **STEP 1: Data passed from user space to kernel**

The write system call issued in Step 4 of B switches processing to kernel space. The kernel then copies the data generated by the user-space program into kernel space and adds it to the send queue associated with the socket specified by the user-space program.

□ **STEP 2: Header processing based on protocol**

In the same kernel context as in Step 1, packet headers are added to the data to be transmitted if necessary. Once the packet is ready and the kernel subsystem determines it is okay to transfer the data, it is passed on to the next subsystem[*1].

□ **STEP 3: Data transmitted from the NIC**

The data (with header added) is ultimately passed to the NIC device driver, and the device driver tells the NIC to send the data.

2.2.2 VM Network I/Os

Now let's look at how communications processing works in a VM environment, with reference to Figure 4.

■ **Basic systm structure**

The four components are (from top to bottom in Figure 4):

(1) Virtual machine (VM)
(2) Virtual NICs assigned to the VM
(3) A host kernel that implements a virtual I/O backend, tap devices, virtual switches, and device drivers
(4) A physical NIC.

VM communication functions can be implemented in a number of ways, but here I consider a format similar to Linux vhost-net, in which threads inside the host kernel function as the virtual I/O backend.

[*1] In some cases, a write system call for a TCP socket (file descriptor) may not immediately result in the data specified by the user-space program being transmitted by the NIC. Possible reasons for this include the TCP implementation's congestion control, Nagle's algorithm waiting for the outgoing buffer to reach a certain size as a means of improving performance, and the delay of data transmissions by subsystems such as qdisc, which handles NIC bandwidth control.

■ Typical loop

On VMs, programs that respond to requests, as discussed above, typically run the following loop: (D) process incoming packets for the VM in the host kernel, Parts (A)–(C) as described above for general-purpose OSes[2], (E) process the VM's outgoing packets in the host kernel.

Packet receipt and transmission processing is summarized in Figures 5 and 6 by execution context.

■ D: Processing incoming packets for the VM

□ STEP 1: Notification from hardware (physical NIC)

The initial processing performed when a packet arrives at the physical NIC is the same as in Step 1 of A above. A hardware interrupt handler is started in the host kernel, and a kernel thread is started to process incoming packets.

□ STEP 2: Incoming packets passed to virtual switch

As in Step 2 of A above, the kernel thread started in Step 1 processes the received packet, but the processing performed is different from in A above. First, the received packet is passed to the virtual switch. The virtual switch reads the Ethernet header of the received packet, finds the appropriate destination interface for the packet, and adds the packet to that interface's receive queue. Here, if the destination interface is a tap device, it starts the backend kernel thread that is responsible for virtual I/O and associated with that tap device.

□ STEP 3: Pass received data to virtual NIC

The virtual I/O backend kernel thread started in Step 2 pulls data from a tap device and pushes it to the virtual NIC's receive queue. It then sends an interrupt to the VM to notify it that packets were received on the virtual NIC.

□ STEP 4: Process incoming packets within the VM

The VM receives the interrupt sent by the host in Step 3, and processing switches to the interrupt handler set up by the kernel within the VM. From this point on, processing within the VM follows the process starting from Step 1 of A above.

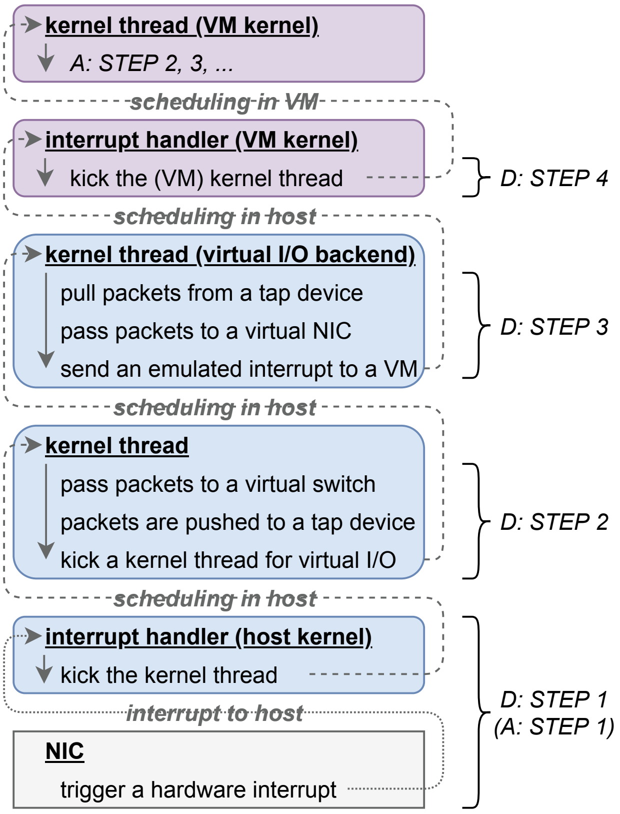
kernel thread (VM kernel)
A: STEP 2, 3, ...
scheduling in VM
interrupt handler (VM kernel)
kick the (VM) kernel thread
D: STEP 4
scheduling in host
kernel thread (virtual I/O backend)
pull packets from a tap device
pass packets to a virtual NIC
send an emulated interrupt to a VM
D: STEP 3
scheduling in host
kernel thread
pass packets to a virtual switch
packets are pushed to a tap device
kick a kernel thread for virtual I/O
D: STEP 2
scheduling in host
interrupt handler (host kernel)
kick the kernel thread
interrupt to host
NIC
trigger a hardware interrupt
D: STEP 1
(A: STEP 1)

Figure 5: Processing of Incoming Packets on a Virtual Machine —Part and Step Indicated at Right

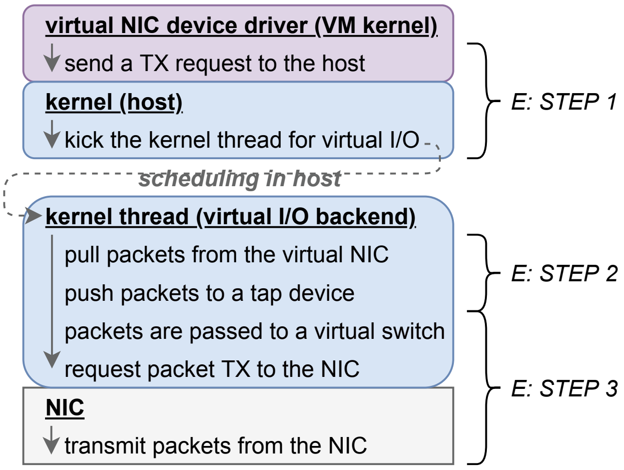
virtual NIC device driver (VM kernel)
send a TX request to the host
kernel (host)
kick the kernel thread for virtual I/O
E: STEP 1
scheduling in host
kernel thread (virtual I/O backend)
pull packets from the virtual NIC
push packets to a tap device
E: STEP 2
packets are passed to a virtual switch
request packet TX to the NIC
NIC
transmit packets from the NIC
E: STEP 3

Figure 6: Processing of Outgoing Packets on a Virtual Machine —Part and Step Indicated at Right

[2] When a general-purpose OS is running on a VM, the behavior of communication programs within the VM is basically the same as described in Section 2.2.1.

■ E: Process the VM's outgoing packets

□ STEP 1: VM sends a transmission request

At this point, the VM has executed Step 3 of C above and added the outgoing data to the virtual NIC's send queue via the virtual NIC device driver. Now when the virtual NIC is asked to send packets, execution context switches from the VM to the host kernel, and the kernel thread for virtual I/O is started.

□ STEP 2: Data passed from virtual NIC to tap device

The kernel thread for virtual I/O started in Step 1 above pulls data from the virtual NIC's send queue and pushes it to a tap device.

□ STEP 3: Transfer data from tap device to virtual switch

After Step 2, packets are passed through the tap device to the virtual switch and transmitted from the interface corresponding to the packet's destination.

2.3 Research Community Efforts

Here, I go over efforts by the research community to speed up the workloads discussed in the previous sections.

2.3.1 Reducing System Call Costs

The faster a NIC's I/O operations, the more frequently (to the extent allowed by CPU resources) the user-space program described in Part B above runs its loop. The point to note here is that the system calls involve switching the user and kernel contexts and are thus CPU-intensive. In specific terms, the workloads discussed in Part B involve frequent system call invocations: select, poll, and epoll _wait in Step 1, read system calls in Step 2, and write system calls in Step 4. The issue is that this increases the amount of time spent on context switching as a proportion of the overall program execution time.

■ Issuing multiple system calls at once

In 2010, researchers presented a system called FlexSC[*3] designed to allow multiple processing requests to be sent to the kernel at once (batching). To achieve this, the system creates a set of memory pages that is shared among user and kernel space. To execute a system call, user-space threads write the system call and its arguments to the shared memory area, and a kernel thread asynchronously executes these calls and returns the results. This mechanism eliminates the need for context switching on a call-by-call basis. Implementation methods differ in their details, but this approach[*4] came to be widely adopted in efforts to optimize network stack implementations, as described in Section 2.3.3.

[*3] Livio Soares and Michael Stumm. 2010. FlexSC: Flexible System Call Scheduling with Exception-Less System Calls. In 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10). (https://www.usenix.org/conference/osdi10/flexsc-flexible-system-call-scheduling-exception-less-system-calls).

[*4] The idea of reducing context switching by issuing multiple requests in batches had been explored before the advent of FlexFC via a technique called multi-calling in compilers[*5] and hypervisors[*6].

[*5] Mohan Rajagopalan, Saumya K. Debray, Matti A. Hiltunen, and Richard D. Schlichting. 2003. Cassyopia: Compiler Assisted System Optimization. In Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9 (HotOS '03), 18. (https://www.usenix.org/conference/hotos-ix/cassyopia-compiler-assisted-system-optimization).

[*6] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the Art of Virtualization. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP '03), 164–177. (https://doi.org/10.1145/945445.945462).

Eliminating the boundary between apps and the kernel

Another approach is to run applications and the OS kernel in the same address space, thus eliminating the boundary between applications and the kernel. This makes it possible for application programs to use features implemented by the kernel via ordinary function calls rather than system calls. This can be done via unikernels[*7], which runs all programs—including applications and the kernel—in the same address space, and library OSes, which implement kernel functions as libraries that can run in user space. In addition to improved performance due to reduced system call context switching costs, unikernels and library OSes also offer other notable advantages such as shorter startup times for high-demand OS functions in data-center environments, reduced memory usage, and improved security. This approach has yielded much research and a range of implementations, including unikernel systems like OSv[*8], IncludeOS[*9], LightVM[*10], HermiTux[*11], Lupin Linux[*12], Unikraft[*13], and Unikernel Linux[*14], as well as library OSes like VirtuOS[*15], Graphene[*16], EbbRT[*17], KylinX[*18], and Demikernel[*19].

[*7] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. 2013. Unikernels: Library Operating Systems for the Cloud. In Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '13), 461–472. (https://doi.org/10.1145/2451116.2451167).

[*8] Avi Kivity, Dor Laor, Glauber Costa, Pekka Enberg, Nadav Har'El, Don Marti, and Vlad Zolotarov. 2014. OSv - Optimizing the Operating System for Virtual Machines. In 2014 USENIX Annual Technical Conference (USENIX ATC 14), 61–72. (https://www.usenix.org/conference/atc14/technical-sessions/presentation/kivity).

[*9] Alfred Bratterud, Alf-Andre Walla, Hårek Haugerud, Paal E. Engelstad, and Kyrre Begnum. 2015. IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services. In 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), 250– 257. (https://doi.org/10.1109/CloudCom.2015.89).

[*10] Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, and Felipe Huici. 2017. My Vm Is Lighter (and Safer) Than Your Container. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17), 218–233. (https://doi.org/10.1145/3132747.3132763).

[*11] Pierre Olivier, Daniel Chiba, Stefan Lankes, Changwoo Min, and Binoy Ravindran. 2019. A Binary-Compatible Unikernel. In Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2019), 59–73. (https://doi.org/10.1145/3313808.3313817).

[*12] Hsuan-Chi Kuo, Dan Williams, Ricardo Koller, and Sibin Mohan. 2020. A Linux in Unikernel Clothing. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys ' 20). (https://doi.org/10.1145/3342195.3387526).

[*13] Simon Kuenzer, Vlad-Andrei Bădoiu, Hugo Lefeuvre, Sharan Santhanam, Alexander Jung, Gaulthier Gain, Cyril Soldani, Costin Lupu, Ştefan Teodorescu, Costi Răducanu, Cristian Banu, Laurent Mathy, Răzvan Deaconescu, Costin Raiciu, and Felipe Huici. 2021. Unikraft: Fast, Specialized Unikernels the Easy Way. In Proceedings of the Sixteenth European Conference on Computer Systems (EuroSys '21), 376–394. (https://doi.org/10.1145/3447786.3456248).

[*14] Ali Raza, Thomas Unger, Matthew Boyd, Eric B Munson, Parul Sohal, Ulrich Drepper, Richard Jones, Daniel Bristot De Oliveira, Larry Woodman, Renato Mancuso, Jonathan Appavoo, and Orran Krieger. 2023. Unikernel Linux (UKL). In Proceedings of the Eighteenth European Conference on Computer Systems (EuroSys '23), 590–605. (https://doi.org/10.1145/3552326.3587458).

[*15] Ruslan Nikolaev and Godmar Back. 2013. VirtuOS: An Operating System with Kernel Virtualization. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13), 116–132. (https://doi.org/10.1145/2517349.2522719).

[*16] Chia-Che Tsai, Kumar Saurabh Arora, Nehal Bandi, Bhushan Jain, William Jannen, Jitin John, Harry A. Kalodner, Vrushali Kulkarni, Daniela Oliveira, and Donald E. Porter. 2014. Cooperation and Security Isolation of Library OSes for Multi-Process Applications. In Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14). (https://doi.org/10.1145/2592798.2592812).

[*17] Dan Schatzberg, James Cadden, Han Dong, Orran Krieger, and Jonathan Appavoo. 2016. EbbRT: A Framework for Building PerApplication Library Operating Systems. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 671– 688. (https://www.usenix.org/conference/osdi16/technical-sessions/presentation/schatzberg).

[*18] Yiming Zhang, Jon Crowcroft, Dongsheng Li, Chengfen Zhang, Huiba Li, Yaozheng Wang, Kai Yu, Yongqiang Xiong, and Guihai Chen. 2018. KylinX: A Dynamic Library Operating System for Simplified and Efficient Cloud Virtualization. In 2018 USENIX Annual Technical Conference (USENIX ATC 18), 173–186. (https://www.usenix.org/conference/atc18/presentation/zhang-yiming).

[*19] Irene Zhang, Amanda Raybuck, Pratyush Patel, Kirk Olynyk, Jacob Nelson, Omar S. Navarro Leija, Ashlie Martinez, Jing Liu, Anna Kornfeld Simpson, Sujay Jayakar, Pedro Henrique Penna, Max Demoulin, Piali Choudhury, and Anirudh Badam. 2021. The Demikernel Datapath OS Architecture for Microsecond-Scale Datacenter Systems. In Proceedings of the ACM Sigops 28th Symposium on Operating Systems Principles (SOSP '21), 195–211. (https://doi.org/10.1145/3477132.3483569).

2.3.2 More Efficient Packet Passing Between User Space and NICs

With 10Gbps NICs now widespread, it has become difficult to achieve wire-rate performance, particularly with small packet sizes, with configurations like that illustrated in Figure 1.

■ **Program behavior**

To address this issue, in the early 2010s researchers presented packet I/O frameworks like Data Plane Development Kit (DPDK)[*20] and netmap[*21] to enable the efficient transfer of data between user space and NICs. Packet I/O frameworks have two basic functions:

(1) Paste the NIC's packet buffer directly into the user-space program.
(2) Provide a lightweight interface to allow the user-space program to
  a) detect new packets received by the NIC, and
  b) request the NIC to transmit packets.

■ **Program behavior**

When a user-space program uses these basic packet I/O framework functions to perform processing in the manner described in Part B above (receiving data and then generating and sending a response), the behavior is as follows.

□ **STEP 1: Detect received packets**

The program uses the interface provided by the packet I/O framework to detect new packets received by the NIC[*22].

□ **STEP 2: Application-specific processing**

As in Step 3 of Part B, the program performs application-specific processing based on the data received.

□ **STEP 3: Transmit data from the NIC**

If the application-specific processing requires data to be transmitted, the program first populates the outgoing packet buffer that was pasted into user space with the data it wants to send, and then uses the interface provided by the packet I/O framework to ask the NIC to transmit the packets.

□ **Caveat**

The overall program behavior described above replaces all of the processing done in Parts A, B, and C in the previous sections and greatly simplifies things by making it possible to pass data between the user-space program and the NIC extremely quickly. But it must be noted that because this does not include protocol-related processing as described in Step 2 of A and Step 2 of C, it is not possible to run a web server that delivers data via TCP connections with this setup as is.

□ **Available cost reductions**

The details depend on the packet I/O framework implementation, but with DPDK[*20], for example, not only is there no intervening protocol-related processing (as discussed in the caveat above), other costs that can be reduced relative to the general-purpose OS environment discussed in Section 2.2.1 include the kernel thread

[*20] Intel. 2010. Data Plane Development Kit. (https://www.dpdk.org/).

[*21] Luigi Rizzo. 2012. Netmap: A Novel Framework for Fast Packet I/O. In 2012 USENIX Annual Technical Conference (USENIX ATC 12), 101–112. (https://www.usenix.org/conference/atc12/technical-sessions/presentation/rizzo).

[*22] When a newly received packet is detected, the data will already be in the packet buffer pasted into the user space, so there is no need to perform the processing described in Step 2 of B.

scheduling that starts in Step 1 of A, the scheduling associated with user-space program startup originating in Step 3 of A, and the system calls and associated copying of memory between user space and the kernel included in Steps 1, 2, and 4 of B.

□ Main use cases

As mentioned in the above caveat, protocol-related processing—such as for TCP—is not performed on data that the user-space program receives from the NIC. This is actually quite useful when network functions such as a router are implemented in software, and so packet I/O frameworks are widely used in contexts like Network Function Virtualization (NFV)[*23]. The research community, for example, has developed packet I/O framework-based NFV platforms such as FastClick[*24], E2[*25], NetBricks[*26], and Metron[*27]. Also, as described in the next section, protocol stacks that run on packet I/O frameworks have been developed to enable applications like web servers to be used in combination with packet I/O frameworks. Packet I/O frameworks are also being used to speed up VM communications, as discussed in Section 2.3.4.

2.3.3 Rethinking Network Stack Design

■ Scaling in multicore environments

Many NICs let you create multiple packet queues to scale performance in multicore environments, and dividing them up for use by separate CPU cores makes it possible to avoid lock contention when attempting to access the queues. Many high-performance NICs also implement a feature called Receive Side Scaling (RSS) in hardware. RSS allows processing to be distributed by steering received packets to specific queues according to TCP connection or IP address. It is not enough to separate the packet queues, however. There is only one queue per socket for newly established TCP connections, and performance does not scale if accept system calls are issued to the same socket in parallel in a multicore environment. To address this, systems such as Affinity-Accept[*28], MegaPipe[*29], and Fastsocket[*30] offer a means of setting up TCP connection queues for each core, and it has been shown that this makes it possible to scale the performance of accept processing in multicore environments. MegaPipe[*29] also enables batch processing inspired by FlexSC[*3], which I covered in Section 2.3.1.

■ Use of packet I/O frameworks

Researchers have studied ways of using packet I/O frameworks to speed up programs like web servers, as mentioned under "Main use cases" in Section 2.3.2. Specifically, protocols like TCP/IP have been implemented that can be incorporated into Step 2 under "Program behavior" in Section 2.3.2, and this makes it possible to eliminate processing costs as mentioned in

[*23] NFV makes it possible to implement network functions in software on commodity hardware, whereas previously you needed to purchase expensive custom hardware appliances for each network function. NFV allows a single computer to be used in multiple applications, and It is considered easier to add/change functions with NFV than with custom hardware. The availability of high-speed NICs at low prices, in particular, has likely been a tailwind for the uptake of NFV.

[*24] Tom Barbette, Cyril Soldani, and Laurent Mathy. 2015. Fast Userspace Packet Processing. In 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 5–16. (https://doi.org/10.1109/ANCS.2015.7110116).

[*25] Shoumik Palkar, Chang Lan, Sangjin Han, Keon Jang, Aurojit Panda, Sylvia Ratnasamy, Luigi Rizzo, and Scott Shenker. 2015. E2: A Framework for NFV Applications. In Proceedings of the 25th Symposium on Operating Systems Principles (SOSP '15), 121–136. (https://doi.org/10.1145/2815400.2815423).

[*26] Aurojit Panda, Sangjin Han, Keon Jang, Melvin Walls, Sylvia Ratnasamy, and Scott Shenker. 2016. NetBricks: Taking the V Out of NFV. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 203–216. (https://www.usenix.org/conference/osdi16/technical-sessions/presentation/panda).

[*27] Georgios P. Katsikas, Tom Barbette, Dejan Kostić, Rebecca Steinert, and Gerald Q. Maguire Jr. 2018. Metron: NFV Service Chains at the True Speed of the Underlying Hardware. In 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), 171–186. (https://www.usenix.org/conference/nsdi18/presentation/katsikas).

[*28] Aleksey Pesterev, Jacob Strauss, Nickolai Zeldovich, and Robert T. Morris. 2012. Improving Network Connection Locality on Multicore Systems. In Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys '12), 337–350. (https://doi.org/10.1145/2168836.2168870).

[*29] Sangjin Han, Scott Marshall, Byung-Gon Chun, and Sylvia Ratnasamy. 2012. MegaPipe: A New Programming Interface for Scalable Network I/O. In 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 135–148. (https://www.usenix.org/conference/osdi12/technical-sessions/presentation/han).

[*30] Xiaofeng Lin, Yu Chen, Xiaodong Li, Junjie Mao, Jiaquan He, Wei Xu, and Yuanchun Shi. 2016. Scalable Kernel TCP Design and Implementation for Short-Lived Connections. In Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16), 339–352. (https://doi.org/10.1145/2872362.2872391).

“Available cost reductions” in that section, resulting in significant speed increases. In 2014, researchers presented user-space network stacks called Sandstorm[*31] and mTCP[*32]. mTCP[*32] offers a number of optimizations. In addition to request batching as proposed in FlexSC[*3], it also divides TCP connection queues among CPU cores as in Affinity-Accept[*28] and MegaPipe[*29], which I mentioned in Section 3.3.1. Also in 2014, researchers presented new OSes called Arrakis[*33] and IX[*34] designed to make it faster to use devices. Both of these allow a network stack based on a TCP/IP implementation called lwIP[*35] to deliver I/O directly to the NIC. In 2016, researchers presented a system called StackMap[*36] that uses a packet I/O framework to adopt the program behavior described in Section 2.3.2 for packet sending/receiving while also offering the benefits of the full-featured TCP/IP implementation in the kernel by using the kernel implementation for TCP/IP protocol-related processing like that in Step 2 of A and Step 2 of C. In 2019, researchers announced a TCP stack implementation called TAS[*37], which also uses DPDK[*20] and operates in user space. In 2022, researchers presented a system called zIO[*39] that extends TAS[*37] and the Strata[*38] file system and makes it possible to eliminate I/O-related memory copying without making changes to existing applications. Researchers have also looked at ways of optimizing the allocation of CPU cores to tasks in order to achieve the low levels of communications latency required in data-center settings, as demonstrated by systems like ZygOS[*40], Shenango[*41], Shinjuku[*42], and Caladan[*43]. These systems also employ a TCP/IP stack running on top of DPDK[*20].

[*31] Ilias Marinos, Robert N. M. Watson, and Mark Handley. 2014. Network Stack Specialization for Performance. In Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM ’14), 175–186. (https://doi.org/10.1145/2619239.2626311).

[*32] EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. 2014. mTCP: A Highly Scalable User-Level TCP Stack for Multicore Systems. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 489–502. (https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/jeong).

[*33] Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe. 2014. Arrakis: The Operating System Is the Control Plane. In 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), 1–16. (https://www.usenix.org/conference/osdi14/technical-sessions/presentation/peter).

[*34] Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. 2014. IX: A Protected Dataplane Operating System for High Throughput and Low Latency. In 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), 49–65. (https://www.usenix.org/conference/osdi14/technical-sessions/presentation/belay).

[*35] Adam Dunkels. 2001. Design and Implementation of the lwIP TCP/IP Stack. Swedish Institute of Computer Science 2, 77.

[*36] Kenichi Yasukata, Michio Honda, Douglas Santry, and Lars Eggert. 2016. StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs. In 2016 USENIX Annual Technical Conference (USENIX ATC 16), 43–56. (https://www.usenix.org/conference/atc16/technical-sessions/presentation/yasukata).

[*37] Antoine Kaufmann, Tim Stamler, Simon Peter, Naveen Kr. Sharma, Arvind Krishnamurthy, and Thomas Anderson. 2019. TAS: TCP Acceleration as an OS Service. In Proceedings of the Fourteenth EuroSys Conference 2019 (EuroSys ’19). (https://doi.org/10.1145/3302424.3303985).

[*38] Youngjin Kwon, Henrique Fingler, Tyler Hunt, Simon Peter, Emmett Witchel, and Thomas Anderson. 2017. Strata: A Cross Media File System. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP ’17), 460–477. (https://doi.org/10.1145/3132747.3132770).

[*39] Timothy Stamler, Deukyeon Hwang, Amanda Raybuck, Wei Zhang, and Simon Peter. 2022. zIO: Accelerating IO-Intensive Applications with Transparent Zero-Copy IO. In 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), 431–445. (https://www.usenix.org/conference/osdi22/presentation/stamler).

[*40] George Prekas, Marios Kogias, and Edouard Bugnion. 2017. ZygOS: Achieving Low Tail Latency for Microsecond-Scale Networked Tasks. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP ’17), 325–341. (https://doi.org/10.1145/3132747.3132780).

[*41] Amy Ousterhout, Joshua Fried, Jonathan Behrens, Adam Belay, and Hari Balakrishnan. 2019. Shenango: Achieving High CPU Efficiency for Latency-Sensitive Datacenter Workloads. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), 361–378. (https://www.usenix.org/conference/nsdi19/presentation/ousterhout).

[*42] Kostis Kaffes, Timothy Chong, Jack Tigar Humphries, Adam Belay, David Mazières, and Christos Kozyrakis. 2019. Shinjuku: Preemptive Scheduling for μsecond-scale Tail Latency. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), 345–360. (https://www.usenix.org/conference/nsdi19/presentation/kaffes).

[*43] Joshua Fried, Zhenyuan Ruan, Amy Ousterhout, and Adam Belay. 2020. Caladan: Mitigating Interference at Microsecond Timescales. In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), 281–297. (https://www.usenix.org/conference/osdi20/presentation/fried).

Offloading processing to hardware

The sort of processing involved in TCP, such as connection state management, is relatively complex, putting high loads on the CPU, so researchers have also explored an approach known as the TCP Offload Engine (TOE) for offloading such processing to hardware devices like NICs. In 2020, researchers presented a system called Accell TCP[*44], which allows processing related to certain states—such as establishing TCP connections—to be offloaded to the NIC, making it possible to perform connection splicing and other such processing at high speed. The researchers showed that this can mainly help improve the performance of L7 load balancers. Also in 2020, researchers presented Tonic[*45], a hardware design that enables the implementation of transport layer protocols in the NIC. In 2022, researchers unveiled FlexTOE[*46], a TOE implementation that runs on smart NICs, and 2023 saw researchers present IO-TCP[*47], a system in which the NIC, in addition to performing TCP processing, is given direct access to storage hardware to streamline content delivery workloads.

2.3.4 Speeding up VM Communications

As Figure 4 shows, the main software components in VM communications are virtual switches that multiplex packet input/output on physical NICs, and a backend that handles virtual NIC emulation. In this section, I go over efforts to optimize these two components.

Speeding up virtual switches

Packet I/O frameworks, which I mentioned in Section 2.3.2, have had a huge impact in speeding up virtual switches, and research has shown that applying packet I/O frameworks around virtual switches can improve performance significantly relative to conventional approaches. In the case of VM I/O as discussed in Section 2.2.2, virtual switches run in Step 2 of D and Step 3 of E, so improving virtual switch performance can be a huge help in enhancing VM communications performance. On the research front, in 2012 researchers presented a virtual switch called VALE[*48] that can run on the netmap[*21] API mentioned in Section 2.3.2, and 2013 saw the unveiling of CuckooSwitch[*49], which uses DPDK[*20]. And in

[*44] YoungGyoun Moon, SeungEon Lee, Muhammad Asim Jamshed, and KyoungSoo Park. 2020. AccelTCP: Accelerating Network Applications with Stateful TCP Offloading. In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), 77–92. (https://www.usenix.org/conference/nsdi20/presentation/moon).

[*45] Mina Tahmasbi Arashloo, Alexey Lavrov, Manya Ghobadi, Jennifer Rexford, David Walker, and David Wentzlaff. 2020. Enabling Programmable Transport Protocols in High-Speed NICs. In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), 93–109. (https://www.usenix.org/conference/nsdi20/presentation/arashloo).

[*46] Rajath Shashidhara, Tim Stamler, Antoine Kaufmann, and Simon Peter. 2022. FlexTOE: Flexible TCP Offload with Fine-Grained Parallelism. In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), 87–102. (https://www.usenix.org/conference/nsdi22/presentation/shashidhara).

[*47] Taehyun Kim, Deondre Martin Ng, Junzhi Gong, Youngjin Kwon, Minlan Yu, and KyoungSoo Park. 2023. Rearchitecting the TCP Stack for I/O-Offloaded Content Delivery. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), 275–292. (https://www.usenix.org/conference/nsdi23/presentation/kim-taehyun).

[*48] Luigi Rizzo and Giuseppe Lettieri. 2012. VALE, a Switched Ethernet for Virtual Machines. In Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12), 61–72. (https://doi.org/10.1145/2413176.2413185).

[*49] Dong Zhou, Bin Fan, Hyeontaek Lim, Michael Kaminsky, and David G. Andersen. 2013. Scalable, High Performance Ethernet Forwarding with CuckooSwitch. In Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '13), 97–108. (https://doi.org/10.1145/2535372.2535379).

2015, researchers presented mSwitch[*50] as an extension of VALE[*48]. Work to add support for DPDK[*20] to Open vSwitch[*51], a widely used virtual switch implementation, was also underway around this time.

■ Improvements to the virtual I/O backend

Around 2013, researchers made attempts to use virtual switches like that described above under "Speeding up virtual switches" in the VM communications backend. One of these attempts uses VALE[*48] as the network backend on QEMU[*52]. Specifically, it replaces the existing virtual switch implementation in the OS kernel shown in Figure 2 with the VALE[*48] switch, and the researchers showed that this can improve VM I/O performance[*53]. With this optimization, however, the virtual NIC assigned to the VM was of the existing type and thus not all that compatible with VALE[*48]. The drawback here was that, in Step 2 of D and Step 2 of E, it was unable to eliminate the copying of packet data memory between the virtual switch and the virtual NIC, so room to improve performance remained. To fill the gap, in 2015 researchers implemented ptnetmap[*54*55], which assigns netmap[*21] interfaces directly to the VM, for QEMU[*52]/KVM[*56], and showed that on ptnetmap, VMs can achieve the 10Gbps wire rate of 14.88Mpps with the smallest possible packet size. In 2014, researchers also presented ClickOS[*57], which uses VMs and replaces netfront/netback, the VM communications mechanism used in Xen[*6], with a communications mechanism based on VALE[*48] and the netmap[*21] API in order to speed up communications throughput on NFV platforms. Also in 2014, researchers presented NetVM[*58], an NFV platform based on VMs, which uses DPDK[*20] and QEMU[*52]/KVM[*56] to speed up VM communications throughput. In 2017, researchers presented a framework called HyperNF[*59], which, even in VM environments that use VALE[*48], addresses the problem of suboptimal CPU utilization due to the separation of the kernel threads running on the host side as described in Step 2 of E and the threads run on the VM's virtual CPU. It does this by performing the sort of virtual switch processing that happens on VALE[*48] within hypercalls that place it inside the virtual CPU's execution context. The researchers showed that the increased efficiency of CPU utilization resulted in high VM communications throughput.

[*50] Michio Honda, Felipe Huici, Giuseppe Lettieri, and Luigi Rizzo. 2015. mSwitch: A Highly-Scalable, Modular Software Switch. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15). (https://doi.org/10.1145/2774993.2775065).

[*51] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. 2015. The Design and Implementation of Open vSwitch. In 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), 117–130. (https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff).

[*52] Fabrice Bellard. 2005. QEMU, a Fast and Portable Dynamic Translator. In 2005 USENIX Annual Technical Conference (USENIX ATC 05), 41–46. (https://www.usenix.org/conference/2005-usenix-annual-technical-conference/qemu-fast-and-portable-dynamic-translator).

[*53] Luigi Rizzo, Giuseppe Lettieri, and Vincenzo Maffione. 2013. Speeding up Packet I/O in Virtual Machines. In Architectures for Networking and Communications Systems, 47–58. (https://doi.org/10.1109/ANCS.2013.6665175).

[*54] Stefano Garzarella, Giuseppe Lettieri, and Luigi Rizzo. 2015. Virtual Device Passthrough for High Speed VM Networking. In 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 99–110. (https://doi.org/10.1109/ANCS.2015.7110124).

[*55] Vincenzo Maffione, Luigi Rizzo, and Giuseppe Lettieri. 2016. Flexible Virtual Machine Networking Using Netmap Passthrough. In 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 1–6. (https://doi.org/10.1109/LANMAN.2016.7548852).

[*56] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. 2007. KVM: the Linux Virtual Machine Monitor. In Proceedings of the 2007 Ottawa Linux Symposium (OLS '07).

[*57] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. 2014. ClickOS and the Art of Network Function Virtualization. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 459–473. (https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/martins).

[*58] Jinho Hwang, K. K. Ramakrishnan, and Timothy Wood. 2014. NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 445–458. (https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/hwang).

[*59] Kenichi Yasukata, Felipe Huici, Vincenzo Maffione, Giuseppe Lettieri, and Michio Honda. 2017. HyperNF: Building a High Performance, High Utilization and Fair NFV Platform. In Proceedings of the 2017 Symposium on Cloud Computing (SoCC '17), 157–169. (https://doi.org/10.1145/3127479.3127489).

■ Offloading processing to hardware

Many NICs implement a hardware packet switching function like Single Root I/O Virtualization (SR-IOV)[*60], and using this can often produce better performance than a virtual switch implementation in software. Yet SR-IOV[*60] only provides limited behavioral control of packet forwarding between physical and virtual interfaces from software, which can impede its usefulness in settings where fine-grained control is needed, such as data centers. To address this, in 2018 researchers published a paper on AccelNet[*61], a system that uses smart NICs to enable more flexible network control (the deployment of AccelNet in commercial environments had apparently begun in 2015).

2.4 Recent Work at IIJ Research Laboratory

In this section, I explain what sort of efforts IIJ Research Laboratory is undertaking based on the past research covered above.

2.4.1 Integrating New OS Features and Existing Programs

As the preceding section illustrates, for over a decade now the research community has been proposing designs and implementations of new OS features that would replace existing mechanisms.

■ Problem

System call hooking is commonly used to apply new OS features transparently to existing application programs.

[*60] PCI-SIG. 2010. Single Root I/O Virtualization and Sharing Specification. (https://pcisig.com/specifications/iov/single_root/).

[*61] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), 51–66. (https://www.usenix.org/conference/nsdi18/presentation/firestone).

But the existing system call hook mechanism does have its drawbacks: it can cause significant application performance degradation, and some system call hooks can fail. These shortcomings limit the applicability of unikernels, library OSes, and new network stack implementations like those discussed in previous sections, which, as a result, prevents many people from enjoying the benefits of the research work that has been done. This problem makes it difficult to use existing technologies that could greatly improve software execution efficiency, which would reduce the number of servers needed and cut power consumption.

■ Solution

To solve this problem, we devised a new system call hook mechanism called zpoline*62*63 that addresses the existing mechanism's drawbacks. zpoline*62*63 replaces the `syscall` and `sysenter` instructions, which are two bytes and used to issue system calls, with `callq *%rax` call instructions (also two bytes) and sets up trampoline code at virtual memory address 0, thus replacing `syscall`/`sysenter` calls in the program with jumps to specific hook processing routines. We found this mechanism to produce loads 28–700x lower than conventional mechanisms, such as binary rewriting techniques that use int3 instructions, ptrace, and Syscall User Dispatch*64. Also, when we used these techniques to apply lwIP*35 and DPDK*20 with Redis*65, a widely used key-value store, we found that, relative to when there is almost no load from system hooks, the conventional mechanisms caused a 72.3–98.8% load degradation vs. only a 5.2% degradation with our proposed method.

2.4.2 Speeding up VM I/O

The communications performance of VMs has improved significantly over the last decade. Yet challenges remain.

■ Problem

The cost of exiting a VM context is considered to be a cause of performance degradation in VM environments

*62 Kenichi Yasukata. 2021. A Method for Rapidly Hooking System Calls with Zero Call Drops. IIJ Engineers Blog. (https://eng-blog.iij.ad.jp/archives/11169, in Japanese).

*63 Kenichi Yasukata, Hajime Tazaki, Pierre-Louis Aublin, and Kenta Ishiguro. 2023. zpoline: a system call hook mechanism based on binary rewriting. In 2023 USENIX Annual Technical Conference (USENIX ATC 23), 293–300. (https://www.usenix.org/conference/atc23/presentation/yasukata).

*64 Gabriel Krisman Bertazi. 2021. Syscall User Dispatch. (https://www.kernel.org/doc/html/latest/admin-guide/syscall-user-dispatch.html).

*65 Salvatore Sanfilippo. 2009. Redis - Remote Dictionary Server. (https://redis.io/).

that has yet to be eliminated. In specific terms, the exit generated in Step 2 of E in Section 2.2.2 hinders I/O performance, which limits the performance of workloads running on the VMs. If the I/O performance of the VM itself is low, this limits the maximum achievable performance even when using the mechanisms described in Sections 2.3.1 and 2.3.3 to streamline processing related to the network running on the VM. And this is a serious issue given that a lot of computational work currently runs on VMs within data centers.

■ Solution

To address this, we developed a mechanism dubbed Exit-Less, Isolated, and Shared Access (ELISA)[*66][*67], which allows the VM to access NICs shared between VMs without exiting the VM context. With our proposed method, the VMFUNC CPU instruction is used to create a new context within the VM in which only behavior permitted by the host can happen, and the NIC can only be accessed within this new context. This prevents VMs from performing malicious behavior through the NIC. Our method also implements methods for VMs to access devices in software, so it can provide greater behavioral flexibility than SR-IOV[*60]. We showed that implementing VM communication functions with our method can yield up to 163% performance gains compared with a mechanism that is similar to HyperNF[*59] in that it exits from the VM context with every VM I/O request.

2.5 Conclusion

I first took a brief look at the general behavior of system software communications functions, and then examined how past research in system software communications since the early 2010s has improved these functions, and then rounded out the discussion with a look at IIJ Research Laboratory’s recent work in this area.

Kenichi Yasukata
Researcher, Research Laboratory, IIJ

[*66] Kenichi Yasukata. 2023. The Path to Getting a Paper Accepted for ASPLOS 2023—Challenges and Solutions in the Sharing of Memory Between VMs. IIJ Engineers Blog. (https://eng-blog.iij.ad.jp/archives/18819, in Japanese).

[*67] Kenichi Yasukata, Hajime Tazaki, and Pierre-Louis Aublin. 2023. Exit-Less, Isolated, and Shared Access for Virtual Machines. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS 2023), 224–237. (https://doi.org/10.1145/3582016.3582042).

3. Focused Research (2)

Evolution of the IIJ Cloud—Commemorating 30 Years

3.1 Introduction

Over IIJ's history, we have provided Internet connectivity alongside a range of related services with a focus on communications. We have continued to enhance and expand the service hosts used to provide these services over the past 30 years, which now encompass several thousand servers. We also provide computing resources to customers through the IIJ GIO cloud service, the infrastructure for which has also grown to comprise tens of thousands of servers in the decade since launch.

In this article, we take a look back at IIJ's 30-year journey. The first half describes how IIJ's service hosts have changed with the times and the innovations we have made in the process. The second half looks at how the IIJ GIO cloud platform has evolved through successive generations to become the large-scale service infrastructure it is today.

3.2 1990s: Where it all Started

The era of dedicated systems for each service

The number of service hosts we deployed to provide email and web services increased rapidly from the time IIJ was founded in 1992. By the end of the 1990s, we had over 200 racks housing several thousand servers distributed across multiple data centers in Tokyo to provide IIJ's services. We maintained separate racks for each service and procured and built the equipment on that basis, which meant that work had to be performed onsite whenever the configuration changed. With this in mind, we sought to optimize server operations by using conveniently located data centers in and around Tokyo that offered easy physical access.

In the early days, we used UNIX workstations as our servers, but we later transitioned to PC/AT-compatible machines and PC/AT-based industrial PCs for cost performance reasons.

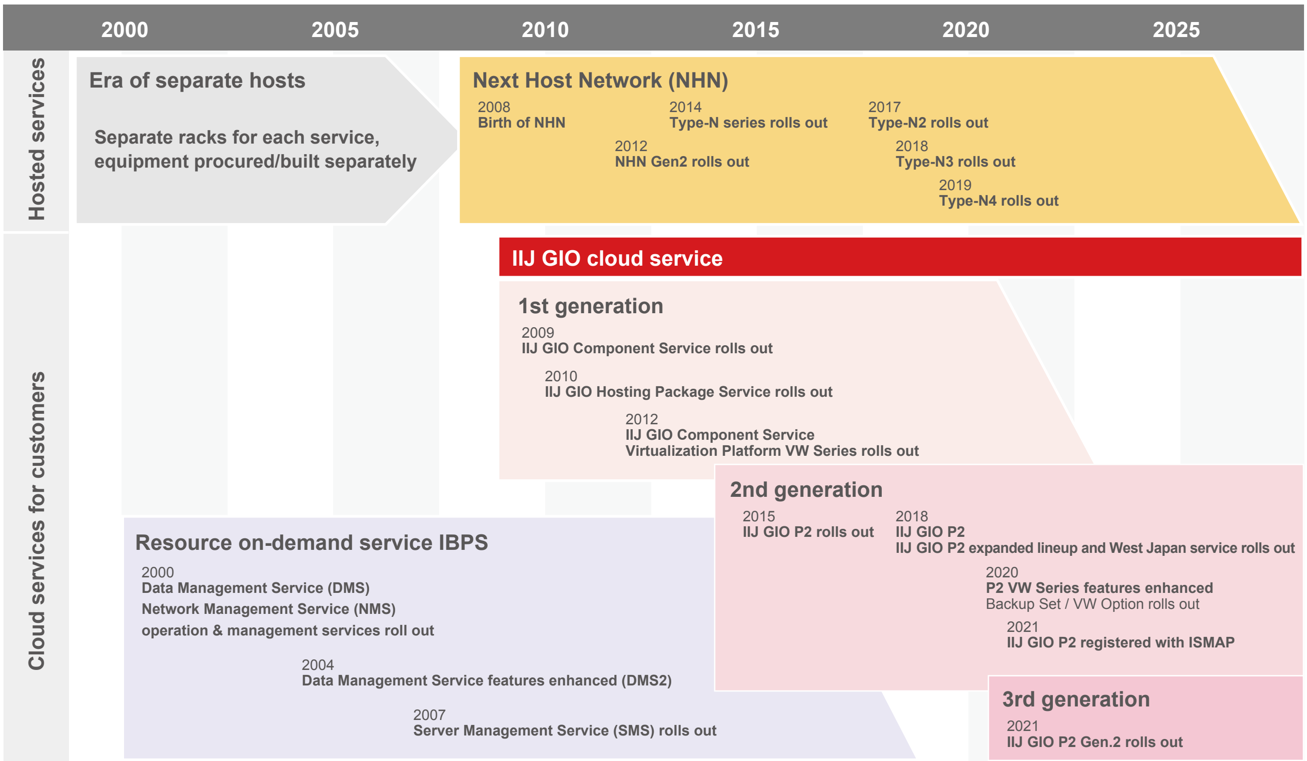
2000
2005
2010
2015
2020
2025
Hosted services
Era of separate hosts
Separate racks for each service, equipment procured/built separately
Next Host Network (NHN)
2008 Birth of NHN
2012 NHN Gen2 rolls out
2014 Type-N series rolls out
2017 Type-N2 rolls out
2018 Type-N3 rolls out
2019 Type-N4 rolls out
Cloud services for customers
IIJ GIO cloud service
1st generation
2009 IIJ GIO Component Service rolls out
2010 IIJ GIO Hosting Package Service rolls out
2012 IIJ GIO Component Service Virtualization Platform VW Series rolls out
2nd generation
2015 IIJ GIO P2 rolls out
2018 IIJ GIO P2 IIJ GIO P2 expanded lineup and West Japan service rolls out
2020 P2 VW Series features enhanced Backup Set / VW Option rolls out
2021 IIJ GIO P2 registered with ISMAP
3rd generation
2021 IIJ GIO P2 Gen.2 rolls out
Resource on-demand service IBPS
2000 Data Management Service (DMS) Network Management Service (NMS) operation & management services roll out
2004 Data Management Service features enhanced (DMS2)
2007 Server Management Service (SMS) rolls out

Figure 1: Evolution of IIJ's Cloud Services

As rack-mounted PC servers were not yet commercially available, IIJ staff often assembled the servers themselves using off-the-shelf parts. A variety of work was performed at the Otemachi data center: we would replace server motherboards and CPUs, IIJ staff would assemble RAID arrays themselves, and we had teams and systems in place to ensure that machines could be replaced immediately if they went down. While we were able to set up our equipment on a low budget, we also encountered more and more physical problems like power supply and cooling fan failures, which can in turn lead to hardware failures.

The OSes installed on commercial UNIX workstations at the time were not all that featureful given the price, and it was difficult to make improvements on our own. And Linux had only just appeared on the scene and was not yet production ready. So we decided to obtain a license for PC-BSD, a derivative of BSD (a collection of software created by a group of developers at the University of California, Berkeley), and reconfigure it for use as a server ourselves. This greatly increased the flexibility of our servers. Both our software and hardware were handcrafted. IIJ was probably the only company in Japan at the time that was quite so particular about the cost and quality of its servers.

This was at a time when the TeleHodai service, which offered fixed-rate phone charges after 11pm, was in widespread use in Japan, and 11pm was thus the peak for personal website traffic. So while everything worked smoothly during the day, once 11pm rolled around, the great influx of web server traffic ramped up server loads, and this would sometimes result in circuit breakers being tripped due to excessive power consumption and entire racks thus losing power.

The mainstream storage options at the time were on-board server storage and, where high capacity was required, DAS, whereby external storage was connected to the servers via SCSI cables. In the 1990s and 2000s, we had to plan and budget around questions like how much physical capacity we needed and how much capacity per server, how much it would cost if we installed some number of servers in a location and configured them in a certain way, which hardware vendor’s servers we should use to make sure we stayed within budget, and where we should locate our data centers. Experiencing major storage failures taught me firsthand about the absolute need for backups and adequate system sizing.

3.3 2000s: Shedding the Dedicated-service-hosts Approach

■ Advent of the Next Host Network

The Next Host Network (NHN) is a platform for IIJ services that we developed in 2008 to streamline service host operations, and it represented a redesign of our server configurations and server operation network. Up to this point, we had procured equipment and built systems on a service-by-service basis, as discussed above, and this meant that overall infrastructure costs were continuously on the rise. Any configuration change triggered the need for onsite work, and the operation of any one system was defined in isolation from the others, resulting in a convoluted setup. And while we had installed sufficient capacity to meet anticipated growth in demand for each individual service, it was difficult to repurpose equipment when our demand forecasts failed to pan out, and maintenance costs were also rising as we generally entered into maintenance contracts on the equipment we selected. The cost of maintaining our physical facilities also continued to soar because we were using conveniently located data centers in urban centers that we could get to quickly whenever system problems occurred.

We needed to fundamentally reconsider the nature of our infrastructure if we were to break away from this chronically high cost structure. This led to the implementation of NHN, designed to solve the myriad problems we faced by consolidating our disparate systems and making it possible to provide an equivalent level of service even while running our operations remotely. The basic concept we adopted was to provide flexible service hosts, and we embarked on a four-year plan to progressively replace thousands of servers distributed across our data centers in Japan, consolidating data that had been stored on DAS systems into iSCSI storage and making this available over an IP network, along with newly installed diskless servers, in a flexible manner according to service demand.

The basic design principle, based on the assumption that servers would inevitably fail, was to redesign storage to be highly reliable and the network to be simple and uncomplicated. We wanted to reduce onsite work and switch to a model that would allow operations to basically be made remote. Server equipment has a wide variety of failure points, and because failures are unavoidable and there are limits to the extent to which failure rates can be reduced, we designed the system so that the inevitable failures would only have a small impact. We moved away from having assigned racks for each service and adopted a server pool system to improve storage efficiency, making it possible to consolidate onsite work and outsource it to contractors. We standardized server configurations to reduce the configuration management workload, and added virtualization technology into the mix to enhance flexibility, enabling post-installation work to be accomplished remotely. Individual HDD failures are inevitable with any storage device, but we needed to adopt a more reliable configuration to avoid failures that would lead to service outages. We used PXE Boot and iSCSI storage with controller and path duplexing to create an inexpensive, highly reliable diskless server environment, making it possible to recover by simply switching server devices in the event of failures. In our experience, edge switch failures had not been all that common, so we adopted a simple configuration that only used NIC redundancy settings and the like where necessary. We introduced a mechanism for automatically rewriting switch VLAN settings based on config data to reconfigure the network without changing the physical wiring, making it possible to reduce configuration errors and operational costs. NHN primarily used x86 servers from hardware vendors with strong cost performance. The only OSes available were Linux distributions likeCentOS, but Windows Server and VMware platforms also later appeared. We selected low-end but highly maintainable storage options to simplify storage maintenance tasks for which we had previously relied on experts, making it easier for IIJ staff to perform maintenance themselves and thus helping to reduce annual maintenance costs. The key characteristics of each NHN generation are outlined below.

NHN Gen1: The first generation of NHN infrastructure, launched in 2008. Gen1 adopted iSCSI storage in order to eliminate DAS, and provided two storage areas: basic (60GB) and expanded (160GB) storage. NHN Gen1 units were configured as a set of four server racks and four storage racks comprising a single unit, but once inventory resources were exhausted, it was impossible to expand in the same L2 area, so it was difficult to adjust inventory on a unit-by-unit basis. Server instances also could not be moved between units. Uplink speed was 1Gbps, and so traffic growth also started to put pressure on upstream bandwidth.

NHN Gen2: The second generation of infrastructure, launched in 2012 and designed to solve the problems of NHN Gen1. The units had the same configuration as Gen1, but rack storage efficiency was doubled (to accommodate 40 units per rack). Characteristically, Gen2 adopted Juniper's Virtual Chassis (VC), had a 10Gbps uplink speed, had racks that would accommodate non-NHN standard service-specific equipment, redundant power supplies, and NIC bonding. However, NHN Gen2 did have a problem in that failures in switches used for top-of-rack (ToR) switching would cause the entire network to be disconnected. The failure would spread to the entire VC unit, resulting in a situation that took quite a bit of time to recover from.

NHN Gen2 Container: A version of NHN optimized for the IZmo/S containerized data-center modules deployed at Matsue Data Center Park. Congruent with the number of racks installed in IZmo/S modules, each unit comprised eight racks and 288 servers (later, 16 of the servers were removed and four storage units added). Although NHN Gen2 Container's slim container dimensions make it space efficient, the narrowness of the aisles inside the containers made maintenance particularly difficult, which caused problems on the operations front.

3.4 2010s – Present: Type-N, a Next-generation Service Platform

■ Shifting to Type-N service hosts

In October 2014, we launched the new Type-N series, the third generation of NHN infrastructure. Until this point, we had provided the servers in bare-metal form, but starting with the Type-N series, we also started providing them as virtual machines (VMs). This meant we could also support reduced spec applications that did not require a full bare-metal server. The characteristics of each Type-N generation are outlined below.

Type-N: NHN's third generation of infrastructure, launched in 2014. Until this point, L2 switches were housed in the router. With Type-N, the L2 switches were housed in the L2 core switch, making it possible to provide the same L2 surface between units. Connectivity with users' racks improved, and inventory resources no longer needed to be adjusted on a unit basis. In addition to ordinary servers, Type-N also made available high memory servers and storage servers equipped with a large number of HDDs. There were problems, however: performance of the switches used for the servers' 1GbE NICs was poor, and the maximum number of connectable devices was low.

Type-N100: A broadband infrastructure option based on Type-N and developed for content streaming services. The name refers to the fact that this option provided 100Gbps infrastructure. Key features were that it used dedicated load balancer units and MPLS and allowed connections to IIJ backbone routers, which handle direct IX connections. Systems had to do diskless PXE booting and run the entire OS in memory (no iSCSI storage provided), and a 1TB SSD was provided for temporary logging and caching purposes.

Type-N2: The fourth generation of NHN infrastructure, launched in 2017. Notably, we did away with the 1G network and went with full 10G all the way to the edge. High-memory servers were set up to support the addition of units for clustered applications based on VMware, and hardware performance enhancements led to improved system consolidation rates. A problem with Type-N2, however, was that storage performance did not meet the requirements of some applications.

Type-N3: The fifth generation of NHN infrastructure, launched in 2018. We adopted what were at the time the latest Intel CPUs and installed flash storage (NVMe SSDs) for a huge jump in I/O performance, and also aimed for significant improvements in network performance. With Type-N3, we did away with the separate basic and expanded storage areas and provided the entire storage area as a single PV (physical volume).

Type-N4: The sixth generation of NHN infrastructure, launched in 2019. We again used the latest Intel CPUs at the time and also adopted new OCP-compliant servers (described below), reducing procurement costs and power consumption. We increased network speeds between the edge and core from 10–20Gbps previously to 40Gbps, and we used chassis switches for the core switches to future-proof against future capacity expansions.

From 2019, we adopted mid-range-class storage systems for our main storage. In 2021, we adopted the successor model of storage (from the same manufacturer) that we had used in NHN's first iteration, naturally for its performance and reliability, and also because, on the operations front, we knew it would fit in nicely with our existing procedures and frameworks. Storage products that support self-maintenance operations, including OS and firmware self-updates, also offer strong cost benefits for NHN, where our basic philosophy is to build and run the systems ourselves.

■ Adoption of OCP servers

The OCP (Open Compute Project), formed in 2011, is an organization that proposes new server standards from hyperscalers like Facebook. Compared with traditional servers primarily developed and produced by manufacturers, the OCP uses a different scheme whereby the cloud service providers that use the servers are heavily involved in the selection, procurement, and manufacturing of the components.

While we were interested in OCP servers, the purchase prices were too high to justify the expense when compared with products from ordinary server manufacturers, and this remained the case up until about 2017. But as 2018 rolled around, with the Japanese yen appreciating and the semiconductor industry experiencing an inventory glut, the price of DIMMs, SSDs, and other such components slipped into decline. And in light of these market conditions, we started to give OCP servers serious consideration. We saw a multitude of issues and concerns: we would be directly exposed to exchange rates; on the hardware front, we would need new OCP-compatible racks and central power supplies, and BIOS and BMC tests had to be performed by the buyer; and on the operations front, maintenance was limited to parts replacements (self-maintenance), and we would have to deal with the ODM vendors in English.

And yet we determined that the benefits would outweigh all these concerns, leading us to the decision to deploy OCP servers. When we actually deployed OCP servers, we saved up to 35% on procurement costs and cut power consumption by up to 30%. For procurement costs in particular, these savings were far beyond the 10% we had anticipated, and this spurred us to expand our deployment of OCP servers. At present, however, there are some enterprise applications and areas to which they are clearly not suited. In the summer of 2019, we made multi-vendor arrangements for OCP servers like we had for existing equipment to diversify the technological and procurement risks we face.

Next, let's look at the evolution of IIJ GIO, our cloud platform for customers.

3.5 2000s: Pioneering IaaS (Infrastructure as a Service)

Launch of the IBPS resource on-demand service

In the late 1990s, the advent of server-side Java and ASP made Internet payment infrastructure a possibility, and thus a lot of e-commerce sites based on dynamic website technology began to appear. The system architecture of e-commerce sites was fairly similar, regardless of how the customer's site was configured, so we realized that it would be more efficient to set up the equipment and resources in advance to have them ready to deploy in the form of system components so that we could provide the necessary combination of those components whenever a customer made a order. With this in mind, we created the resource on-demand service IBPS (Integration & Business Platform Service)[*1], the predecessor to our present-day cloud service, IIJ GIO.

IBPS was launched in March 2000 by then group company IIJ Technology Inc. (absorbed into IIJ in April 2010). The service offered everything an Internet business would need, from server equipment through to software, payment/logistics components, and monitoring, operations, and management. The necessary components were combined in accord with the customer's needs to form a complete system, which was provided as an outsourced service. This made it possible to more than halve (relative to IIJ's conventional offerings) development times and initial investments and meet the demands of companies looking to rapidly deploy Internet businesses.

IBPS's four main services

- Data Management Service (DMS): IIJ's storage resources provided on-demand. DMS optimized storage costs by providing storage that allowed the connection type (NAS/SAN), capacity, and performance to be selected according to the customer's service level and usage patterns. DMS also provided high-speed backups on separate disks, freeing customers from the need to perform the old labor- and time-intensive tape backups.
- Network Management Service (NMS): IIJ's network resources provided on-demand. NMS pooled load balancers, firewalls, etc. and provided them as functions. Typical configurations of these were also provided as low-cost, packaged options.
- Server Management Service (SMS): IIJ's server resources provided on-demand. SMS employed a provisioning tool to manage customer-specific server configurations, thus semi-automating the server building task and making systems tailored to customer-specific needs available with minimal lead times.
- Operations and Management Service: This service provided operation and monitoring of customer systems built on IBPS to help reduce total cost of ownership.

Customers were able to use the resources they needed when they needed them, and cancel resources they no longer needed. Users were freed from the risk of ownership. IBPS was what we would now call an IaaS offering. We originally had UNIX servers (SPARC Solaris) on the frontend, and although we later migrated to x86 servers, UNIX servers remained on the middle/backend for database purposes. We had been using rack-mounted servers, but in August 2007, we introduced blade servers—a first for large-scale server infrastructure in Japan—for SMS, through which we offered the first resource on-demand servers in Japan. We liked the ability to save space with blade servers, and the fact that you simply had to insert the CPU blade into a pre-configured chassis to get them working.

[*1] The service was called iBPS at launch but rebranded to IBPS in 2002.

© Internet Initiative Japan Inc.

In October 2003, we expanded the Data Management Service we had been providing since launching IBPS and added a large-scale storage service with a total capacity of 40TB. For midrange storage, we adopted storage virtualization software to enable flexible resource management and reduce costs. In 2007, we upgraded the Data Management Service, halving the per-GB storage costs[*2]. The Data Management Service originally provided SAN and NAS options, but as services offering high-quality Fiber Channel (FC) at reasonable prices became available, the use of FC-SAN storage grew. On the NAS front, we initially used a combination of commodity servers and cluster software or dedicated NAS storage appliances. Few of the NAS products around in the early 2000s supported volumes over 1TB, so if you wanted large-capacity NAS, you would build a NAS system by combining commodity servers, FC-SAN storage, and cluster software. In the late 2000s, large-volume support started coming to NAS products, prompting a shift toward NAS-specific storage appliances.

■ 2008 onward: L2 ring protocol

From 2008, we used an L2 ring protocol configuration for our service platform network technology. The multi-stage ring physical configuration was suited to the scale of data centers in Japan. This had the advantage of making it possible to build an efficient system while also making it easy to expand the system, whether at the edge or the core. With node interfaces of 10GbE+, however, there were fewer models of equipment to choose from than with 1GbE or lower, so a disadvantage was that this limited the number of accommodated nodes. Many L2 ring configurations are manufacturer-specific, and they are not inter-compatible. The problem with this was that it made vendor lock-in likely and thus limited the range of available equipment options.

■ Storage array systems for a variety of platforms

In terms of service platform storage technology, we have used storage array systems since launching IBPS in 2000.

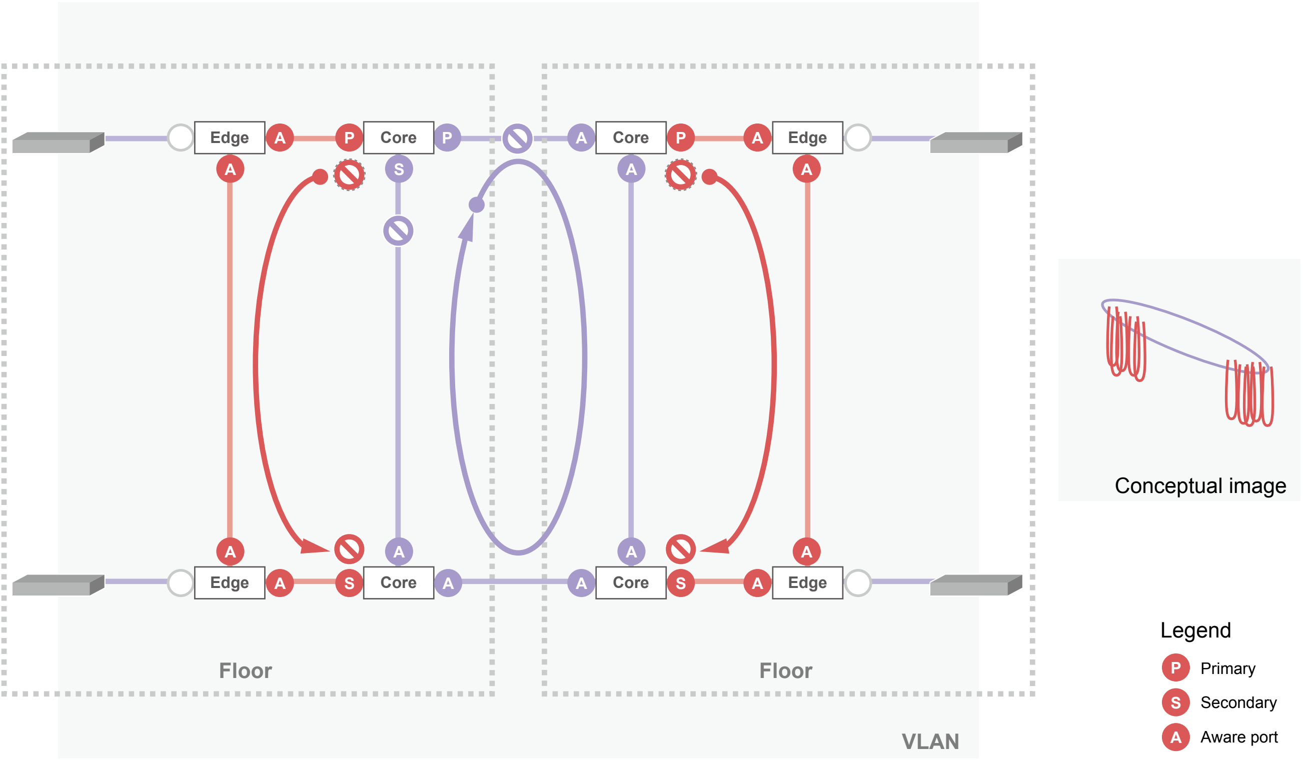

Figure 2: L2 Ring Protocol Configuration (2008–)

[*2] This was also at a time when Moore's Law worked well.

We used several types of mainly high-end – mid-range storage systems to suit performance requirements, and also to diversify procurement and technology risks. Today, we also use storage array systems on the IIJ GIO and NHN platforms, with storage capacity continuing to increase.

When first introducing a storage option, we would contract all of the configuration changes and monitoring out to the vendor, partly because of our initial lack of knowledge. But this style of operations resulted in a fair amount of cash outlay and lead time just to create, for example, a single LUN, so IIJ staff learned how to operate the storage products so that we could make storage configuration changes in-house, which reduced operating costs. Initially, we used IIJ's standard monitoring service to monitor storage, but as the service continued to grow, the inevitable faults started to overwhelm this standard service, and it ended up taking longer for the storage operations team to recognize faults whenever they occurred. We therefore built a dedicated monitoring system for our storage equipment.

3.6 2010s: Cloud Computing Takes Off

■ Birth of the IIJ GIO cloud service

We have overcome a myriad of challenges through our experience in the IaaS business and the evolution of our services platforms spanning more than a decade, including the deployment of virtualization and provisioning systems in IBPS services, the development of in-house monitoring systems, and large-scale upgrades to server, network, and storage equipment. This process ultimately culminated in the development of the IIJ GIO cloud service, which embodies everything we learned. IIJ GIO harnesses the advantages of cloud computing to make high-service-level system environments with high availability and strong security available to users who need business-ready infrastructure at a lower cost, without the need for them to own IT assets themselves.

Our first step was the November 2009 launch of the IIJ GIO Component Service, a private cloud service geared to the fine-grained needs of business customers. This was followed by the June 2010 launch of the IIJ GIO Hosting Package Service, an inexpensive public cloud service with packaged options.

■ IIJ GIO Hosting Package Service

The IIJ GIO Hosting Package Service makes it easy for users to build information systems infrastructure for e-commerce sites and high-performance online businesses simply by selecting pre-packaged plans online according to the needs of their application. The service architecture makes full use of virtualization technology to provide the flexibility to select server resources according to system requirements, and control functionality developed by IIJ makes it possible to automatically allocate multiple server resources on a Layer 2 network. We also adopted open source software (OSS) and developed in-house provisioning tools for controlling resource allocations and management tools for automating complex operations and monitoring, leading to substantial efficiency gains on both the operations and cost fronts. By consolidating our infrastructure, we were also able to reduce hardware procurement costs and set affordable service prices, starting from 8,000 yen per month for a virtual server. In September 2013, we released an IaaS API to meet the needs of users wanting to create their own programs to automate routine tasks, such as the deployment of multiple virtual servers at once.

■ IIJ GIO Component Service

The IIJ GIO Component Service provides servers, storage, and networks in the form of system "components" to create a highly flexible IaaS-based enterprise cloud service that allows users to combine the best components for their needs from a variety of options. The service can also be used as a private cloud by connecting it directly to the user's on-premises environment via a wide area network. The main components of IIJ the GIO Component Service are the base servers and, launched in August 2012, the Virtualization Platform VW Series (the VW Series). Two

© Internet Initiative Japan Inc.

types of base servers were made available: V Series virtual servers that allow users to share physical servers with other customers, and X Series physical servers that give users exclusive use of server hardware units. With the VW series, we provided customers with a private environment with VMware vSphere ESXi (US-based VMWare's virtualization software) on the physical servers along with a VMware vCenter Server management server, all with administrator privileges. This provided a level of flexibility in systems configuration comparable to that with on-premises environments, ensuring it could be used with confidence not only by those looking to integrate servers or build a cloud from scratch, but also by users who already maintained and operated virtual infrastructure using VMware.

We also provided functions other than server resources as IaaS offerings. The first is the Network Add-on feature, which enhances the network functions provided as standard on base servers and the VW Series. Switching from a shared Internet connection line to a dedicated line (private connection) lets users safely connect to their on-premises environment via an Internet VPN or closed network (wide area network). They can also use multi-carrier configurations that divide WAN lines among carriers. The second is the Storage Add-on feature, which broadly offers two options: Standard, which provides high-end storage of the sort that financial institutions use, and Basic, a mid-range offering suitable for data management on ordinary web systems and the like. These storage options are provided as NAS, FC-SAN, or iSCSI-SAN storage over the network. The third is the Database Add-on feature, which provides Oracle Database and MySQL as DBaaS (DataBase as a Service) offerings. When we launched this feature in July 2012, we became the first operator in Japan to offer Oracle Database—which commands a dominant share of the Japanese database market—for a monthly fee. We made it possible for users to lower their initial investment in database licenses, reduce maintenance costs, and avoided investment risks. With this service, IIJ draws on its extensive experience installing and operating relational databases to design and operate database instances and make them available on IIJ GIO virtual servers. We also made it possible to connect to existing on-premises environments via a closed network or Internet VPN. In May 2014, we revised our pricing to reduce monthly fees by up to 56%, and we continued to actively develop the services, adding Microsoft SQL Server to our lineup in October of the same year. The fourth is the License Add-on feature, which provides software licenses for use on the cloud for a monthly fee. We offer Microsoft SPLA licensing as well as licenses for in-demand products and services from vendors like Red Hat, VMware, Arcserve, and Trend Micro.

In January 2014, we bolstered the IIJ GIO Component Service lineup to add Base Server V Series G2 (V Series G2), which built on and enhanced the existing Base Server V Series. V Series G2 was compatible with Windows Server 2012 R2, then the latest version, and compared with the previous series' Windows lineup, offered increased CPU, memory, and disk capacity, and had up to triple the CPU performance at 24 ICU[*2] and up to six times the memory at a maximum of 48GB. We also made the server equipment available at sites in both East and West Japan, so it was viable for disaster recovery applications as well.

■ Release of IIJ GIO Infrastructure P2, our 2nd generation cloud service

In October 2015, we overhauled IIJ GIO's IaaS offerings and launched a new lineup under the IIJ GIO Infrastructure P2 (IIJ GIO P2) banner. Up to this point in our IaaS lineup, we had provided the public cloud IIJ GIO Hosting Package Service, which users could easily deploy online, and the bespoke IIJ GIO Component Service, which allowed users to combine a variety of IT resources to configure a complete system. With IIJ GIO P2, however, our aim was to provide a service covering the full gamut of user needs by combining improved public cloud reliability and processing performance with private cloud offerings that can be requested and deployed immediately online. IIJ GIO P2 comprises public resources

[*3] ICU is a measure of CPU performance. 24 ICU is equivalent to six cores x 2.

offering shared resources primarily through virtual servers, private resources offering dedicated VMware virtual environments and physical servers, and storage resources available on all servers in both the public and private resources offerings. Users can select the optimal combination of resources to build their systems. IIJ GIO P2 also offers excellent external connectivity, with multi-carrier support and private segment extensibility. It makes it possible for users to seamlessly integrate their own on-premises systems and third-party cloud service environments.

Public resources: These are shared resources primarily available on virtual servers, constituting a public cloud adaptable to a wide variety of use cases—from development environments and simple web services to platforms for online games and e-commerce sites requiring high I/O performance. Users can choose the best server resources for their needs from three characteristic types.

- Performance guarantee type: For users that require stable processing performance, this resource type provides virtual servers to which CPU resources will always be reliably allocated. It offers peace of mind for a fixed monthly fee.
- Best effort type: This resource type uses CPU cycle distribution to provide low-cost virtual servers. Pricing is based on usage in one-hour increments so that users can optimize cost outlays.
- Dedicated type: This type provides dedicated virtual servers to users that require high I/O performance capable of withstanding heavy workloads. These virtual servers run in secure server environments physically separated from other users' servers and are equipped with SSDs or SanDisk high-speed flash storage.

Users can combine the three server types and switch among them at will via an online interface. They can also capture OS images from running virtual servers and keep them in a dedicated storage area, enabling the rapid deployment of new virtual servers based on those images. This reduces operational workloads, and enables rapid scaling out during times of heavy workloads and rapid patch deployment. Pricing for the custom OS image storage area is based on usage, so users avoid unnecessary costs.

Private resources: A highly reliable private cloud, suitable even for mission-critical systems, to which business customers can easily migrate systems built in on-premises environments. We offer a lineup of user-dedicated resources with a focus on VMware virtual environments and physical servers. Users can also request the resources using an online interface, something that was not possible with the IIJ GIO Component Service. Via the control panel, users can access services instantly (on the standard model) and self-manage their server resources, adjusting the amount they need in daily increments. IIJ GIO P2 offers enhanced performance specs compared with the IIJ GIO Component Service. Users can select CPUs supporting up to 24 cores, 192GB of memory, and network bandwidth of 10Gbps. Disk and other server specs can also be customized online. By increasing installed memory capacity to facilitate greater server consolidation, we made it possible for users to design and build servers to their system requirements themselves.

■ Expanded IIJ GIO P2 lineup and West Japan launch

In June 2018, we enhanced the performance of the Virtualization Platform VW Series (P2 VW Series) servers provided as part of IIJ GIO P2, adding VW48-1024-FC-10G—which doubled the number of available CPU cores to 48—and VW96-1024-FC-10G—equipped with 96 cores. We released the former on June 1, 2018, and the latter in October 2018. The added options were geared toward anticipated demand for data-processing infrastructure with high core counts and high memory capacity for applications such as AI information processing and SAP S/4 HANA.

Also in June 2018, we launched IIJ GIO P2 public resources in the West Japan region, adding private resources and storage resources in October that year. The East and West Japan regions are connected by a private backbone service provided by IIJ, and the inter-regional broadband network is made available free of charge. This expansion also broadened the range of viable use cases. For example, users could now configure systems based on their business continuity plans (BCP) by using services across sufficiently geographically separate regions.

In July 2020, we enhanced the P2 VW Series to add the Backup Set / VW Option, facilitating easy, cost-effective VM backups. The Backup Set / VW Option uses RCDM (Rubrik Cloud Data Management) to provide the components needed to back up VMs in a single package. With this option, users can simply select the required plan from the backup settings menu to enable easy backup and restore operations. The acquired data is encrypted and stored on backup servers in IIJ's cloud, eliminating the need for users to build and operate backup systems, thus reducing costs and workloads.

■ From L2 MLAG to L2 over L3

From 2014, we used L2 MLAG as our service platform's network technology. An advantage of this is that decent scale can be achieved even with 10GbE+ node interfaces. MLAG is implemented by the data center switch manufacturer, so both the hardware and software are optimized for data centers, but it must be noted that the designs are not suited to all types of facilities. They are basically designed for fairly large floor areas, so depending on the area of the floor on which you intend to install your equipment, as well as power supply and cooling capacity, it might not be possible to install enough equipment to use up all switch ports. So unless you can design and configure the floor to use chassis (assuming the number of accommodated nodes is determined), it's easy to see how you can get into a situation in which you can't use the equipment efficiently.

From 2017, we adopted an L2 over L3 configuration. While this offers the same benefits as MLAG, its strength lies in being able to design a flexible physical topology using L3 technology, which has a long and proven track record on the Internet. The VxLAN implementation at that time had design constraints, however. The standard implementation was not mature enough to enable dynamic control of, for instance, which VLAN each VTEP is configured for, so our design positioned the VTEPs in locations that would allow them to realistically be managed with static settings.

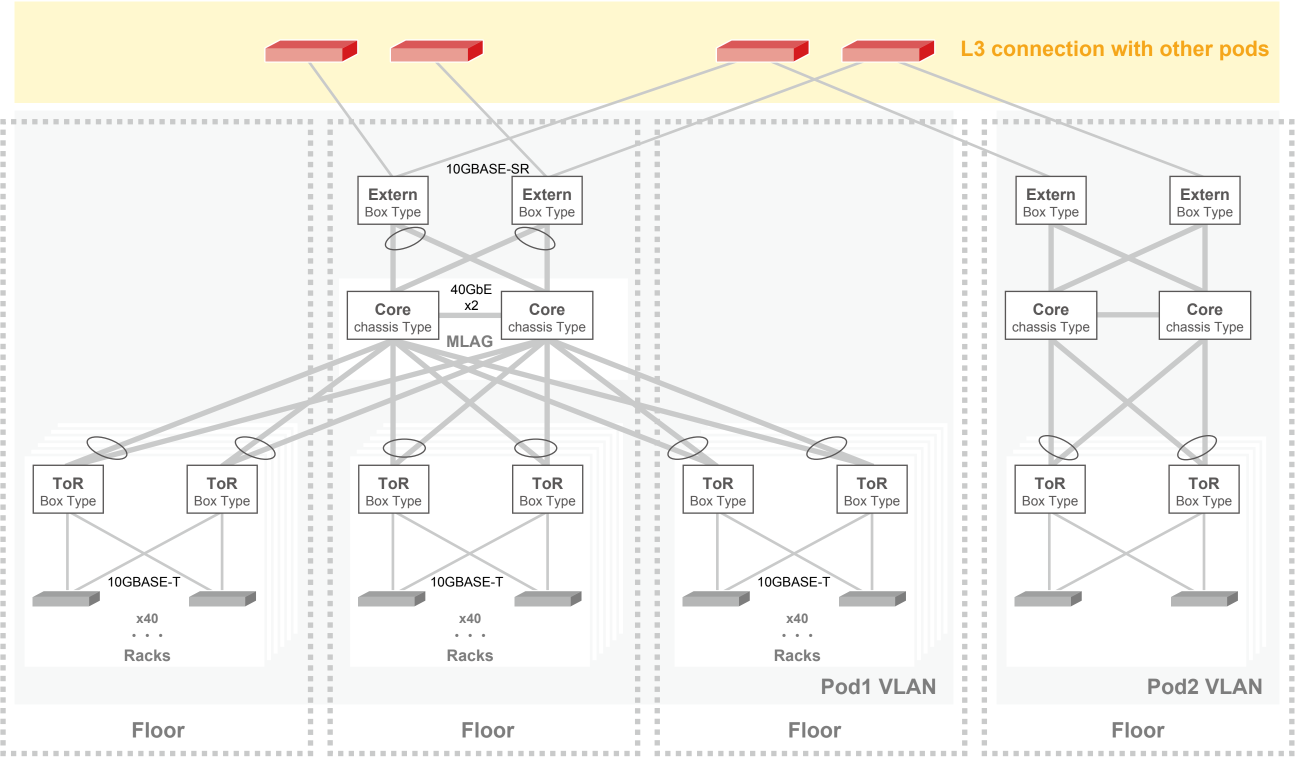

Figure 3: L2 MLAG Configuration (2014–)

Automating storage configuration

Until around 2010, whenever a user requested storage on IIJ GIO, the engineer responsible for running the storage had to manually change the storage and FC switch settings. Up to this point, that only happened infrequently, at most once a week, so even with the engineers performing these changes manually, we were quite capable of fulfilling the lead times set in the service specifications. After 2010, however, that once a week turned into several times a week, and then several times a day, and we could tell it was going to be difficult to sustain things manually, so we designed and built a system to completely automate storage configuration changes.

Storage settings can easily be automated with commercially available applications, but many such applications offer more than just storage configuration automation features (and are thus expensive), so we created the storage control system ourselves. Storage control is performed using scripting languages like Python and Ruby. When using these sorts of languages to manage storage configurations, it is generally preferable to do it via the Storage Management Initiative Specification (SMI-S, a storage management standard) or a proprietary API. The storage and FC switches we were using at the time did not support SMI-S or an API out of the box and instead required separate (expensive) applications, so we used the command line applications that came as standard with the storage products. The input and output capabilities of such command line applications were not really created with humans in mind, and the output in particular was often in a format that is extremely difficult to handle with a scripting language, so the programs you create have to do a lot of cumbersome string processing. Plus, with some of the devices we used, the scripts would return a code of 0 (completed successfully) even when the command or parameters contained an error, so we had to add our own error handling.

The most time-consuming task is that of updating the firmware on storage array systems and FC switches. With some products, such updates can change command line output, so we needed to perform tests in a development environment before upgrading in production. More recent storage products and FC switches increasingly offer support for configuration management software like Ansible, so there has been progress on device API implementations,

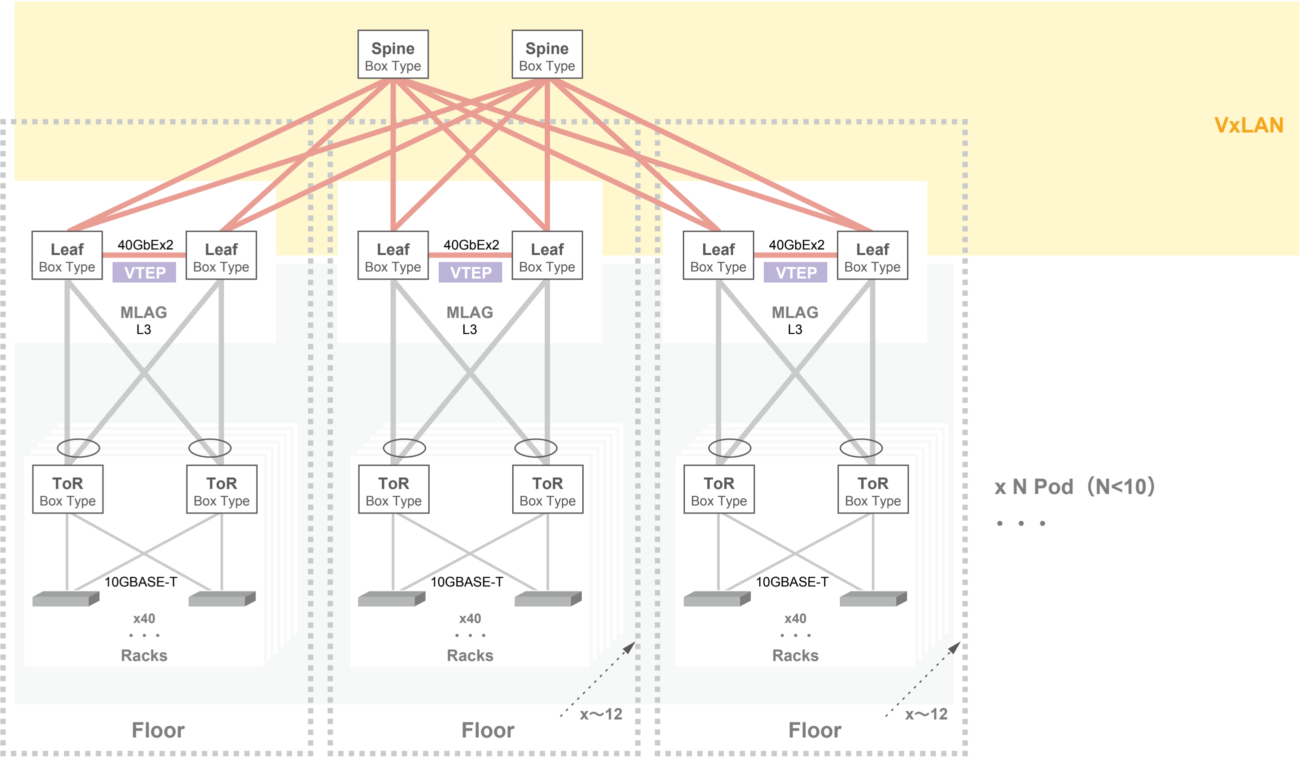

Figure 4: L2 over L3 Configuration (2017–)

© Internet Initiative Japan Inc.

and vendors now offer modules that can be used to manage storage using Python and the like, which has made the programming task much easier than it used to be. From 2010 onward, we have continued to follow the procurement policies we used for previous generations, but as performance requirements increase and the range of storage array products meeting our selection criteria grows, the number of different storage products we use is also increasing.

3.7 2020s – Present: Ongoing Evolution

■ Launch of IIJ GIO Infrastructure P2 Gen.2

On October 1, 2021, we launched IIJ GIO Infrastructure P2 Gen.2 (GIO P2 Gen.2), our next-generation IaaS model, which fully integrates the public IaaS and private IaaS offerings developed and provided under the IIJ GIO brand and makes it easy to migrate systems from on-premises environments.

A key characteristic of GIO P2 Gen.2 is that it uses VMware as its virtual infrastructure and allows the design concepts and operational systems of on-premises environments to be migrated as is, and thus, like GIO P2, continues to target demand for the migration of systems from on-premises VMware environments to the cloud. As a successor to GIO P2, it is also naturally designed to serve as a potential new home for existing users. With GIO P2 Gen.2, users are free to create VMs with a minimum of 1vCPU / 4GB memory from a virtual resource pool, instead of on a per-server basis like in ordinary private clouds. In other words, users can migrate the machine specs that they run in their current environment to GIO P2 Gen.2 as is. This setup means that users can migrate physical machines and VMs from their existing environment by using images or performing P2V or V2V migration. In addition to providing a range of managed services with components that are essential in corporate IT environments, such as file servers, Active Directory, and databases, GIO P2 Gen.2 also abstracts out the hypervisors and hardware such as servers, storage, and networks, so users need not worry about differences between devices. This greatly reduces the user workloads that came with the GIO Virtualization Platform VW Series, where users had to perform software updates to deal with vulnerabilities arising from them managing their hypervisors, and migration tasks in the case of hardware upgrades. Linking GIO P2 Gen.2 with IIJ's other various services, such as network services that provide closed connections to third-party public clouds, also makes it possible to use the platform in anticipation of

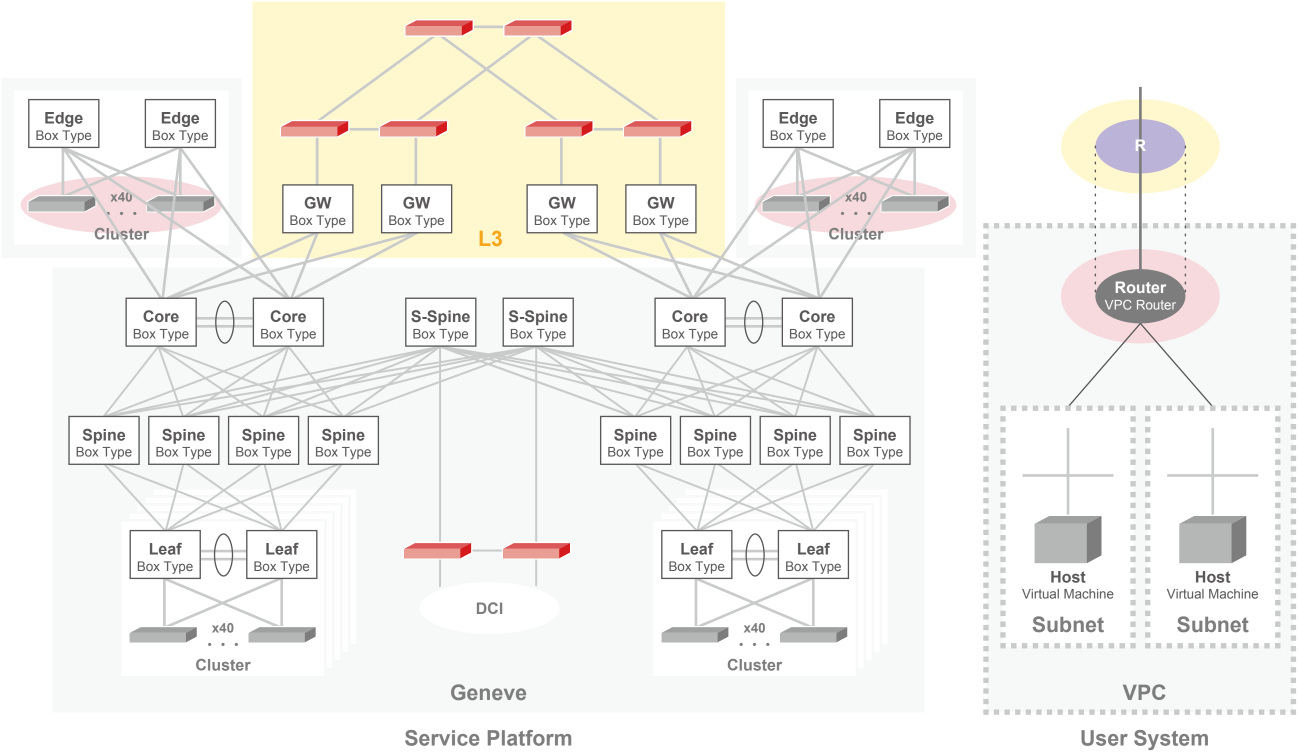

Figure 5: Complete Separate of the Control and Data Planes (2021–)

a future migration to a public cloud. In addition to providing Flexible Server Resources (FSR), which let users freely select resources and configure their system environment just like with a private cloud, we also continue to provide Dedicated Server Resources (DSR) equivalent to what was available on the P2 VW Series using VMware vCenter Server with full privileges.

For GIO P2 Gen.2's Flexible Server Resources, we use VMware Cloud Director (VCD), a product for service providers from VMWare, and hide the hypervisor (vSphere) layer from the user. This lets us provide users with flexible resource control privileges on par with vSphere while allowing IIJ as the service provider to manage the hypervisor and hardware lifecycles. This newly defined shared responsibility model makes it possible for IIJ to manage and operate the hypervisor network. We also provide migration functionality as a service feature, allowing customers to migrate to the cloud with minimal downtime and operational overhead. To operate the hypervisor layer network efficiently, we adopted VMware NSX-T Data Center (NSX-T), which allows comprehensive integration with vSphere, and significantly improved the IaaS network. GIO P2 Gen.2 uses NSX-T to configure an overlay network on top of a Layer 3 IP fabric underlay network, completely separating out the network for each tenant and providing it as a VPC (virtual private cloud). Combining this design with the knowledge we gained from operating a large-scale server pool with IIJ GIO has made it possible for us to allocate resources to users unfettered by the actual physical placement of computing resources (CPUs, memory, storage).

■ Overlay networks using SDN technology

The minimum design requirements for IaaS platforms that accommodate multiple users are physical resource sharing that ensures resources are deployed efficiently and inter-user security. The products on the market offering virtualization technologies for servers as a user computing resource were sufficiently mature, but given the need to ensure interoperability with existing protocols, the implementation of network virtualization is something that has only moved ahead with considerable caution. In recent years, with the improvement of hardware performance and the development of SDN technology, network virtualization technology has become a viable option in large-scale networks. We recognized the usefulness of overlay networks, a type of network virtualization technology, from an early stage, and we have used them through the GIO platform generations, selecting reliable technologies in each instance.

To ensure performance, we previously used network hardware functions to achieve overlay network termination, but with GIO P2 Gen.2, we switched to doing this using the virtual switches within the hosts. This, of course, ensured complete separation between users and also enabled the complete separation of user systems and the physical infrastructure systems. This reduced the impact of underlying system changes on user systems, making it easier to add new functionality.

The loose coupling with the physical layer is also advantageous when deploying IaaS across multiple sites. As with the existing GIO system, not only can equipment be deployed on a large scale across a small number of locations in East and West Japan and seamlessly connected, we also expect it to be easy to distribute equipment on smaller scales across various locations as a disaster preparedness measure.

On the other hand, whereas the settings for each device were previously managed separately due to hardware limitations, the shift to processing in software removed these limitations, resulting in a truly huge amount of configuration information. Since this exceeds what a human can possibly grasp, to ensure quality, we need to move away from the conventional practice of having humans manually make changes. And to ensure services are available to users in a timely manner, we also need to move away from static device configuration and enable dynamic configuration. To achieve these goals, we are using an orchestrator (based on a commercial product with missing or additional features developed in-house) to centrally manage configurations. Centralizing in this manner makes inter-system processing available to linked services via APIs as well, and makes it possible to start related services safely and quickly.

3.8 Conclusion

We've taken a look back at IIJ's 30-year history through a service infrastructure lens. The service host infrastructure underpinning IIJ's various services continues to evolve as we seek to balance stability and efficiency while keeping an eye on the relentless march of innovation and adopting the best technologies to keep up with the times. We mark IIJ GIO's 24th anniversary this year, and the platform's service lineup continues to expand to meet diverse business infrastructure needs. With demand for digital social infrastructure based on AI technology growing in recent years, we are working to provide ultra-high-density AI computing platforms to make this a reality. At IIJ, we will continue to develop and provide services and fundamental technologies to serve market needs.

Shinri Kimura

Head of Cloud Services Division 2, Cloud Division, and Director of Technology Development, Cloud Division, IIJ
Since February 2001, Mr. Kimura has been engaged in the development and operation of large-scale service and cloud infrastructure, and the provision of cloud services. He is also responsible for teams engaged in technology development, human resources development, etc.

December 2023 Vol.60

Internet Initiative Japan

About Internet Initiative Japan Inc. (IIJ)

IIJ was established in 1992, mainly by a group of engineers who had been involved in research and development activities related to the Internet, under the concept of promoting the widespread use of the Internet in Japan.

IIJ currently operates one of the largest Internet backbones in Japan, manages Internet infrastructures, and provides comprehensive high-quality system environments (including Internet access, systems integration, and outsourcing services, etc.) to high-end business users including the government and other public offices and financial institutions.

In addition, IIJ actively shares knowledge accumulated through service development and Internet backbone operation, and is making efforts to expand the Internet used as a social infrastructure.

The copyright of this document remains in Internet Initiative Japan Inc. ("IIJ") and the document is protected under the Copyright Law of Japan and treaty provisions. You are prohibited to reproduce, modify, or make the public transmission of or otherwise whole or a part of this document without IIJ's prior written permission. Although the content of this document is paid careful attention to, IIJ does not warrant the accuracy and usefulness of the information in this document.

©Internet Initiative Japan Inc. All rights reserved.
IIJ-MKTG020-0058

Internet Initiative Japan Inc.

Address: Iidabashi Grand Bloom, 2-10-2 Fujimi, Chiyoda-ku,
Tokyo 102-0071, Japan
Email: info@iij.ad.jp URL: https://www.iij.ad.jp/en/