Designing a Large-scale Email System

3.1 Introduction

In April 2017, we conducted a full system overhaul for IIJ Secure MX Service (SMX), which was launched in October 2006. The SMX service's primary focus is on providing email gateway functionality. Before customers' systems receive emails, the emails first pass through IIJ's email servers, where email-borne threats are filtered out using a variety of technologies, including virus filtering, spam filtering, sender authentication filtering, backscatter filtering, and sandbox filtering. The safe emails are then delivered to customers' email systems.

Over 10 years have passed since launch, and the landscape for emails services has changed substantially over that time, in terms of the volume and size of emails that pass through as well as other factors, from server specs through to email system requirements. SMX has seen system expansions on top of expansions to cope with those changes and thus now has a vastly different makeup from the system that it started out as. In many cases, however, fundamental aspects of system architecture can be traced all the way back to launch, and we have thus felt for quite a while now that the system has its limitations.

In our recent overhaul, therefore, we revised the entire system from the architecture up. This report describes the design of the SMX email system, particularly its delivery system, along with some background to the revisions.

3.2 Challenges and Goals of the System Overhaul

To begin with, we set a number of goals in view of issues with the old delivery system.

3.2.1 Review of the Architecture

The first goal was to revise the old architecture, which had been expanded excessively. Classic large-scale email systems commonly use architectures that line up simple mail servers (message transfer agent, MTA) in series as shown in Figure 1. Pre-overhaul, SMX's delivery system also used a similar architecture.

The biggest benefit of multistage MTA architectures like this is the high extensibility. Delivery system functionality can be expanded easily by linking in MTAs with the desired additional functionality. Also, because the MTAs are connected via SMTP, the standard protocol for email transmissions, linking in products from different vendors does not raise any concerns in terms of interface compatibility, and it is generally rare for product mixing and matching to cause any problems.

This method of expanding a system, however, comes with side effects and should thus not be overdone. The biggest side effect of concern is the increases in storage I/O and operating costs associated with adding more MTAs. In multistage MTA delivery systems, received data is written to storage every time an email passes through an MTA,

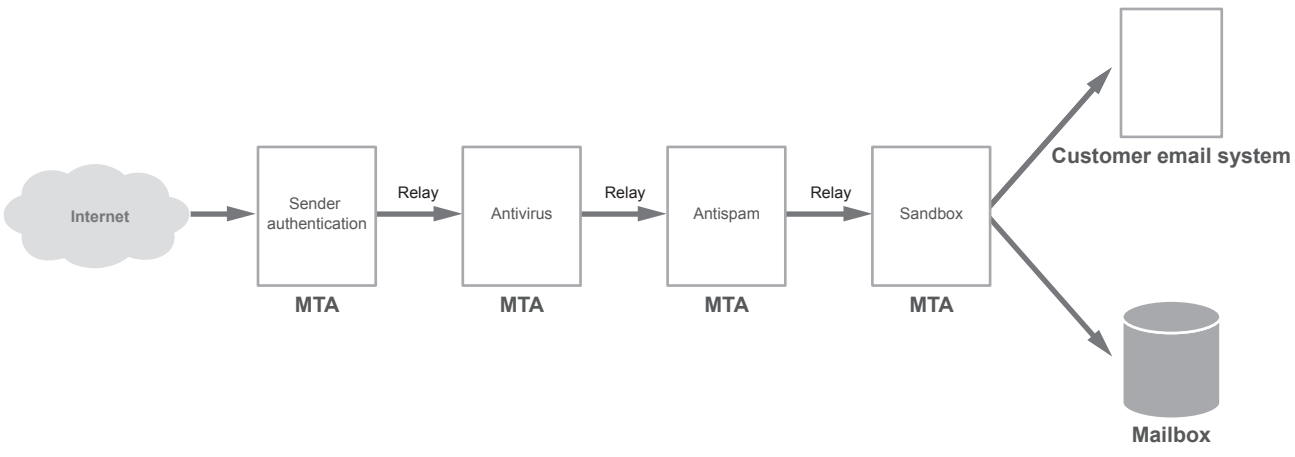

Figure 1: Architecture of a Classical Email System

resulting in storage I/O volumes several times that of the email size. Emails with virtually the same content are written to storage over and over, so taken as a whole, this sort of delivery system architecture results in a plethora of unnecessary storage I/O operations. When network storage solutions such as NAS and SAN are included, these I/O operations alone put a strain on the storage network, creating a bottleneck for the delivery system as a whole. Not only was storage a bottleneck for the old SMX delivery system, the system was also unable to take full advantage of advances in hardware speed and capacity because CPU and memory resources were being left idle.

Also, lining up different MTAs in series means filling the delivery system with a mix of various MTAs that operate in different ways. Not only does this increase the knowledge required—in terms of, e.g., how to view the logs, perform tasks, and deal with problems, and the build procedures to follow when expanding the delivery system—each MTA can also create forks in the delivery route, send out notification emails, and so on, which complicates the flow of emails within the delivery system and makes it tough to understand the delivery system as a whole.

With SMX, the delivery system was repeatedly expanded as scale increased, making for a complicated and expensive delivery system, further expansion of which had become difficult.

3.2.2 Better Filtering

The second goal was to improve the accuracy of virus filtering (antivirus) and spam filtering (antispam). Antivirus and antispam functions are crucial and constitute the dual centerpiece of email security services.

While many security vendors provide both antivirus and antispam services and products, the security industry is subject to rapid change; new attack methods constantly arise, and vendors are constantly developing new technologies to counter them. What this means is that Vendor A's product may offer high detection accuracy on one day, while Vendor B's service may offer high detection accuracy on another, and then the situation may suddenly change when Vendor C releases a new product offering superb accuracy. Put differently, this also means that sticking with any one vendor's engine comes with the risk of declining detection accuracy as the technology used by the engine becomes obsolete. In view of this, we felt we needed a mechanism for keeping virus and spam email detection accuracy high.

3.2.3 Avoiding Over-reliance on Any One Vendor

The third goal was to avoid over-reliance on any specific vendor. To provide a wide range of functionality, the SMX system incorporates numerous vendor products and services. Although many of the products and services offered are appealing, to avoid vendor lock-in, we felt we needed to retain control over how reliant we are on any single offering.

It is not uncommon, particularly among foreign vendors, for a company to be suddenly bought out by a competitor, resulting in its products and services being terminated. Suddenly eliminating the ability to use a product has no small impact on customers who incorporate it into their systems. Although we cannot reduce the impact of this to zero, we do need to take steps to minimize it.

3.3 Single-stage MTA for Effective Use of Hardware Resources

In overhauling the delivery system in line with the above goals, we first sought to harness the plentiful CPU and memory resources afforded by the increasingly high performance of commodity server hardware and to reduce storage I/O, which was a bottleneck for the system overall as well as a factor in costs.

The architecture we arrived at is the polar opposite of the previous one. In short, all processing will be completed within a single, multifunctional MTA, with this single-stage setup eliminating the unnecessary relaying of emails between MTAs (Figure 2). With this architecture, email is only written to storage once, so storage I/O operations are greatly reduced compared with the old delivery system, which wrote the same content to storage multiple times.

Also in the old delivery system, the storage I/O bottleneck meant that the CPU resources of most servers were left idle while CPU utilization rates were high on only some servers, where high-CPU-load tasks such as virus scans were being run. By using identically configured MTAs in parallel, previously idle CPU resources can be diverted for high-load processing tasks, making it possible to more efficiently use CPU resources across the entire delivery system (Figure 3).

Because SMX employs multiple antivirus and antispam engines, the single-stage MTA architecture requires several engines to be loaded into any one server's memory. In general, antivirus and antispam engines hold a lot of data in memory and thus tend to consume a lot of memory resources. Loading several such engines into memory necessitates the sort of total memory capacity that older servers could not accommodate, but the increasingly high performance of commodity server hardware , as mentioned above, has made this sort of setup possible.

3.4 Making Antivirus/Antispam Engines Interchangeable

Next, we designed the overall system to allow the antivirus and antispam engines to be replaceable at any time.

IIJ does not develop antivirus or antispam engines in-house but instead provides this functionality by incorporating such engines from security vendors into the delivery system. This means that IIJ cannot directly address any problems that arise with detection accuracy. Taking the opposite perspective, however, by taking advantage of the ability to cut obsolete technology loose and swiftly incorporate fresh technology, we aimed to design a system that is not tightly tied to any specific antivirus or antispam engine.

First, we evaluated the antivirus and antispam engines of each security vendor. We ran scans for viruses and spam on emails received by honeypots run by IIJ, accumulating

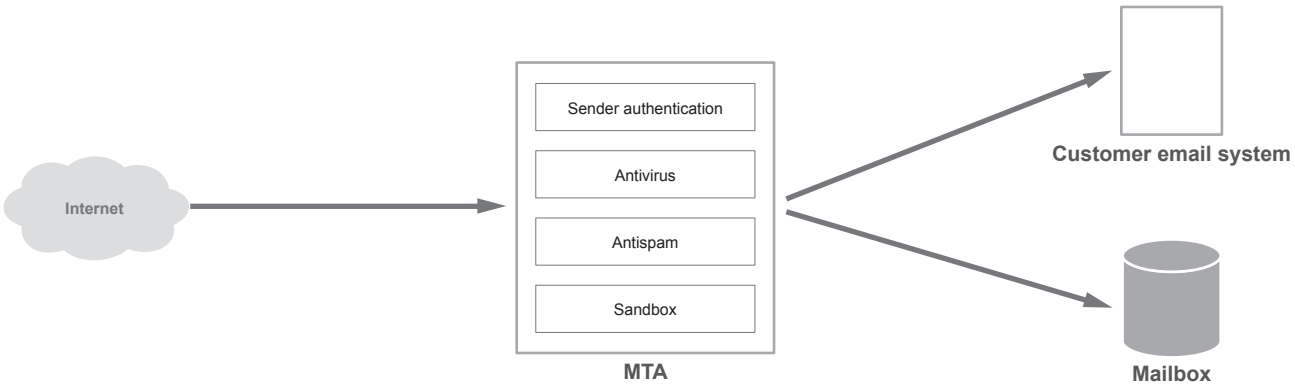

Figure 2: Architecture of the Overhauled Email System

statistics on detection performance over several months to facilitate comparisons. Virus and spam emails can exhibit epidemic-like qualities, so any evaluation of countermeasures over short periods can be heavily influenced by detection performance with respect to whatever spam campaign*1 happened to be active at the time, and thus may not provide a proper idea of long-term detection performance. This is why we used a fairly long validation period. During the validation period, we received product guidance from a range of security vendors. The approaches taken for antispam engines, in particular, are quite distinct across the different products available, and the results of our validation exercise are very interesting in that they reflect the differences in approach. It was an arduous process, but it gave us an understanding of the detection accuracy and tendencies of each engine.

To achieve enhanced detection accuracy, we combine, respectively, multiple antivirus engines and multiple antispam engines. We then combine the results given by each to arrive at the final detection results.

It is evident from the results of this validation process, and also simply from intuition gained through day-to-day operations, that no single engine is an out-and-out winner in terms of how long it takes to detect virus and spam emails after they begin to circulate. Engine A may be quicker to detect such emails during one campaign, while Engine B may be quicker during another, and so on. Combining multiple engines enables us to reduce detection misses during the early stages of a campaign. In the case of antispam engines, in particular, we combine engines that use different approaches so that the weaknesses of any one engine are compensated for by other engines.

Our main aim in combining multiple engines was to improve detection accuracy, but it also had secondary effects. One is the reduction of scan errors. It is not uncommon for virus and spam scans to fail to complete successfully because of corrupted email headers/attachments or deliberate content manipulation, for instance. Running scans with multiple engines makes it possible to significantly reduce the number of emails that cannot be scanned at all. Also, in rare cases, antivirus engines and antispam engines can crash when scanning specific emails or attachments. In such cases, you have the option of disconnecting the engine experiencing the problem as an emergency measure to allow emails to actually make it through the system. It also makes it possible to minimize the impact of a security vendor being bought out and its products becoming unavailable.

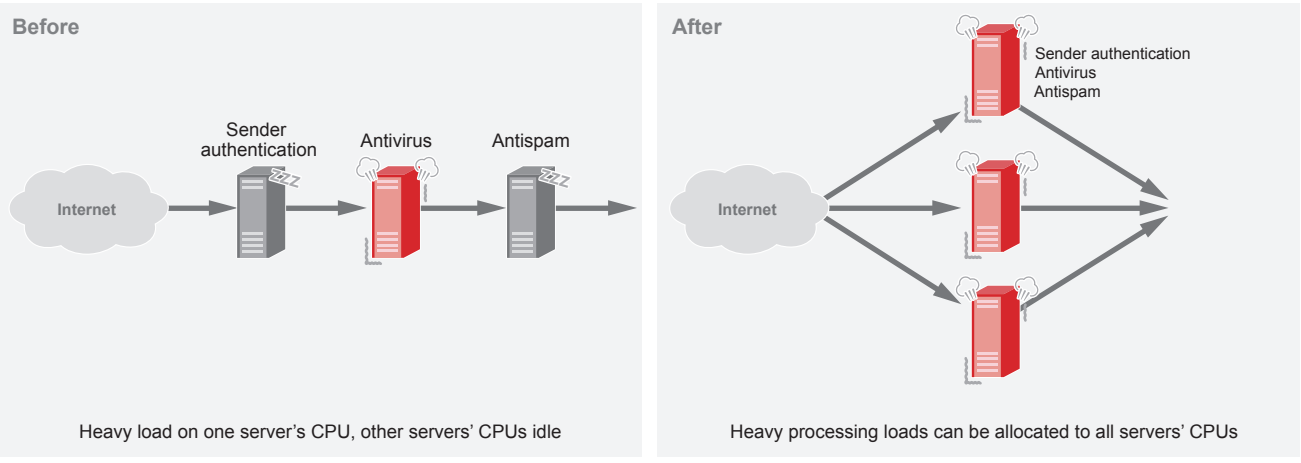

Figure 3: Effective Use of Hardware Resources

*1 Mass mailout of identical or similar spam emails.

3.5 Decision to Develop MTA In-house

The biggest problem we had when overhauling the delivery system was how to implement this design. This is what prompted our decision to develop the MTA in-house, but other methods were available to us, such as combining open source MTAs like Postfix and Sendmail, or the use of MTAs made by MTA vendors.

Postfix and Sendmail provide an interface called Milter, which provides advanced functionality and makes it possible to easily and safely implement email control and rewrite capabilities. On the other hand, given the architecture, adding Milter onto the system would result in a large I/O overhead in particular. And we also had to admit that it lacks the functionality for realizing a complex system like SMX. Although one idea could be to modify Postfix or Sendmail directly, the cost of keeping up with updates to the official package is far greater than you might imagine.

The option of adopting a vendor-produced MTA seemed like a very realistic one. Several MTA vendors develop MTA products for ISPs and large-scale senders. In these products as well, the idea of performing all processing in a single stage, as described above, is the mainstream approach. They also allow various security vendors' antivirus and antispam engines to be swapped in and out, and they allow flexible, extremely fine-grained customizations that meet the detailed requirements of large-scale delivery systems, and as such, MTAs from vendors are far more versatile and powerful than the open-source offerings. For us, this would have been the quickest and easiest way of creating the delivery system we had in mind. Indeed, many ISPs and large-scale senders use vendor MTAs, and parts of SMX's old delivery system also incorporated vendor MTAs.

The only, and biggest, concern we had with taking on a vendor MTA was that SMX would become one with that particular MTA. In a single-stage MTA architecture, the MTA itself is the delivery system. And the entire system beyond the delivery system, from the service specifications through to the operation procedures, would depend heavily on the MTA. In other words, what the MTA can do, SMX can do; and vice versa.

IIJ continues to actively expand SMX's functionality with a close eye not only on customer feedback but also on the latest trends in the email and security industries. So the SMX delivery system may at times require functionality that other operators do not require. In such situations, getting a vendor to add features that an MTA lacks is generally difficult. MTA vendors serve many customers, so they will inevitably put priority on developing functionality that many customers require, and on functionality that key customers want. Developing highly idiosyncratic and niche functionality will naturally be of low priority. Vendor MTAs would appear to be suitable when the required specifications are clear and when few specification changes are likely occur in future, but whether such offerings could support our proactive approach to expanding SMX is an unclear.

Vendor MTAs also come with the risk of the vendor being bought out. Because the system as a whole is heavily reliant on the MTA, were such a corporate acquisition to occur, the impact would be immense relative to any impact that might result in relation to the antivirus and antispam engines. Indeed, several acquisitions involving MTA vendors and products have taken place in the past few years. In absolute terms, not many vendors develop MTAs, so in percentage terms, this is a risk that cannot be ignored.

The final option is in-house development, but this is also not an easy option. MTAs that support the huge levels of communication flows seen at the ISP level require extremely high levels of stability, robustness, and performance. Such a system would also need to provide the diverse functionality and flexibility of a system like SMX. And the technical capabilities to support future expansions in functionality would also be needed.

The decision to develop such an MTA from scratch in-house is perhaps not a very easy one. IIJ, however, does have experience and knowhow in developing many email system components itself. It also has a battery of reliable development teams, and this is why we were able to pull the trigger on developing our system in-house—the risk was high, but we also saw that we had much to gain.

3.6 Overhaul Outcomes

3.6.1 Achieving our Development Goals

The overhaul project encompassed the entire system, including the MTA component. More than a full year passed before the first release was out. It was the biggest development project I have ever been part of.

A lengthy development period and repeated testing were needed to complete the system, but ultimately it met all of the goals we set. Our flexible, versatile MTA made it possible to create a delivery system with a single-stage MTA architecture. In accord with our design, we reduced storage I/O operations, which had been an issue with the old system, thereby enhancing the performance of the delivery system overall. Major reviews of the virus filters and spam filters also resulted in improved detection accuracy. And constantly keeping an eye on the detection rates of each engine we use has also enabled us to take swift action whenever detection rates change.

Developing an MTA in-house has freed us from the risk of MTA vendors being acquired, and it has also enabled us to minimize the effect of security vendors being acquired because we are now able to swap engines in and out. Although perhaps quite obvious, I think the most important outcome of this system overhaul is that we have laid foundations that will enable IIJ to run its own services as it sees fit, essentially immune to the vicissitudes of vendor acquisitions.

3.6.2 Secondary Benefits

We also realized some secondary benefits beyond the goals that we originally set.

First, troubleshooting speed improved. When problems occur with a vendor MTA, details of the condition are reported to the vendor and a fix requested. But if it is unclear how to reproduce the problem, or if the details cannot be passed to the vendor because they contain customer data, then it may take considerable time for the vendor to confirm the problem or, as is often the case, the vendor may be unable to ascertain what the problem is at all. With in-house development, however, the operations and development teams can work closely with one another. This means that the root cause of any problems, in particular, can be identified extremely quickly, and proper provisional and permanent responses implemented.

Also, and while this is not something that affords direct comparisons, I feel that developing the system in-house resulted in strong motivation levels for the team. When developing a system that incorporates MTAs or other products from vendors, the development team develops ways of interfacing with those vendor products, but this frequently involves a seemingly futile struggle against unclear product specifications and is often not much fun on a personal level. In-house development, on the other hand, entails a far, far greater amount of work, and we were concerned about the burden this would place on the development team. Yet, while the team was certainly busy, I never felt an atmosphere of fatigue or exhaustion setting in. In the end, I think there is something exciting about building a large-scale system on your own and seeing it gradually come together.

We released the new system in April 2017, and we spent a full year migrating over to it from the old one. We have received various opinions and requests since release, prompting us to add functions and fix issues, and I think we have been able to so swiftly make these improvements precisely because we developed the system in-house.

I would note that my intention is not to criticize systems that use vendor products across the board. Both approaches have their advantages and disadvantages, so you need to select the right balance depending on the situation. Finally, I would also note that SMX incorporates many vendor products, including some from foreign vendors, into its system, and we work closely with vendors on a daily basis as we provide our services to customers.

Here, I have described the design of the SMX delivery system. With its newly acquired architecture, we will continue to evolve the SMX service going forward.

Takahiko Suzuki
Senior Engineer, Service Development Section, Application Service Department, Network Division, IIJ
Since joining IIJ in 2004, Mr. Suzuki has continued to work on email service development.
He is the developer of yenma, an open-source sender authentication filtering program.