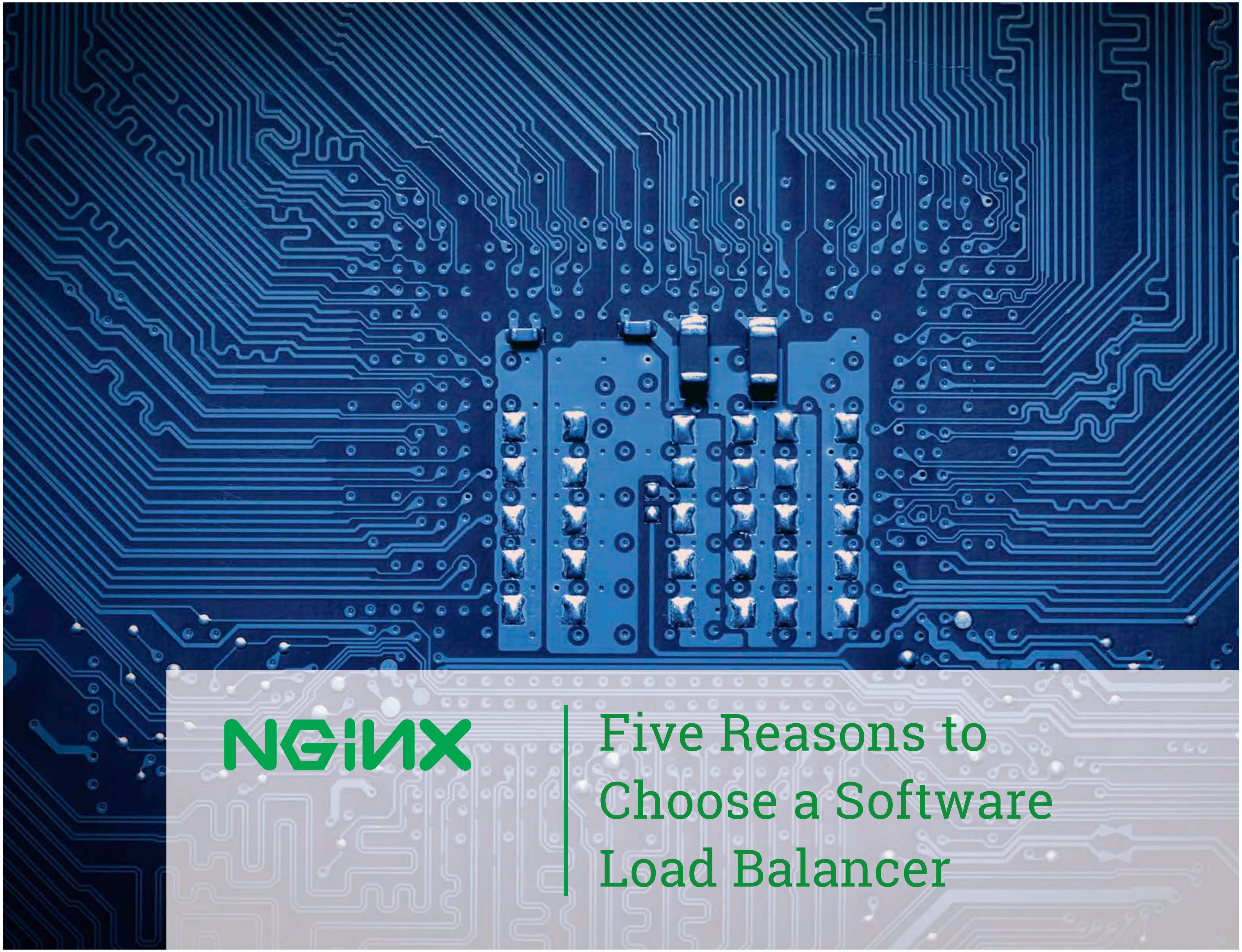
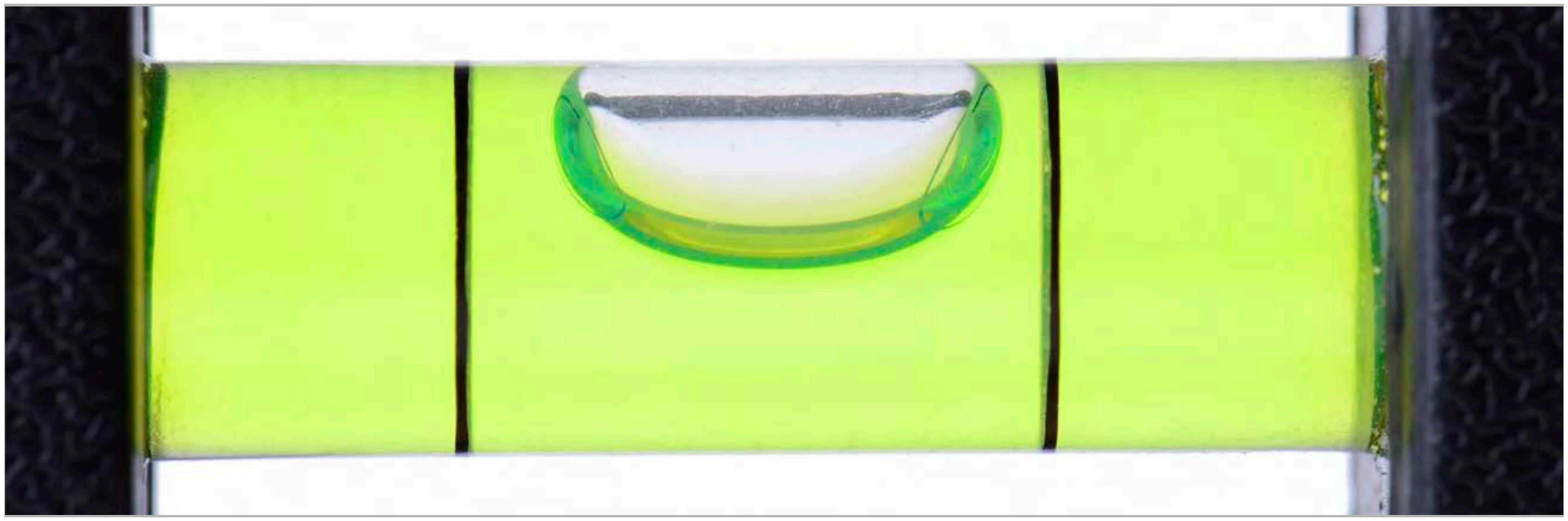


NGINX

# Five Reasons to Choose a Software Load Balancer



# Five Reasons to Choose a Software Load Balancer

*Once upon a time, load balancing in most IT architectures was heavily dependent on hardware—but that's all changed. Companies have been steadily moving away from hardware wherever possible and moving instead to Software-as-a-Service and virtualized or cloud-based infrastructures.*

By now, we’re all familiar with the reasons for heading this direction. It’s less expensive, it’s more scalable, it’s easier to maintain, and—perhaps most importantly—it provides companies with a more flexible development environment. This added agility is critical to application teams because it enables them to complete faster development cycles and concentrate on turning out better features and keeping up with demand.

The problem, however, is that features and functions are meaningless if applications can’t perform. And, for companies still relying on hardware-based legacy Application Delivery Controllers (ADCs) or load balancers, performance could be a serious issue.

In the following pages, we’ll explain why hardware-based load balancers can be a hindrance to your ability to deliver applications. We’ll also present five reasons why software-based load balancing—like that provided by NGINX—might be the best news for your applications *and* your business at large.

## Clearing the hardware hurdle

Applications rule the world now. They aren't just tools that run our workplaces—they're how we run our *lives*. And the more we rely upon our applications, the more we crave immediate response. When features are lacking or buggy, we want them fixed right away. And we also want applications that load fast, at all times, and on any device.

Many companies have understood this urgency *in part*. They have adopted more agile development processes, for example, so developers can incrementally refine and add to application feature sets. And, in support of this development approach, they have migrated much of their hardware infrastructures to the cloud. Cloud-based or virtual development environments bred new software-oriented data centers, which in turn led to cloud-based or virtual delivery mechanisms, in order to satisfy user demand for anytime/anywhere access.

Companies made these moves because they realized that hardware could be limiting in many ways:

- It's expensive. High costs drain budgets and lead to procurement hassles and vendor limitations.
- It requires specific expertise. Implementation and configuration complexity require management by an infrastructure team whose objectives are frequently at odds with those of the application development team.
- It takes time to get going. Reliance on the infrastructure team delays deployment.
- It's rigid. Scalability is tedious and can require extra components, which in turn increase costs, complexity, and deployment times.

But there's a lingering component to this equation that many businesses have forgotten or neglected: ***application performance.***

The reality is that how applications perform is just as important as what they do. After all, if apps don't perform, and users go elsewhere to meet their needs, all development effort is wasted. It makes no sense, then, for businesses to abandon their development hardware and deliver applications online, yet still hang on to a hardware component (ironically) designed to elevate application performance. In fact, it's completely contradictory—like going to the trouble of raising a thoroughbred racehorse and then never letting it out of the starting gate.

Why? Because hardware-based ADCs or load balancers pose many of the same challenges as other hardware components. They're limited and monolithic, and not designed for cloud-based or virtual environments. They also typically fall under the same infrastructure team that handles networking, servers, and so on. As such, they are acquired as part of the infrastructure budget, and they are costly to purchase, maintain, and scale. That expense, along with complexity of configuration, means that each device is often tasked with providing ADC services for a very large number of applications (also known as "multi-tenancy"). Granted, it's not commercially practical to devote a single hardware ADC to each application—but there's a tradeoff in ADC configurations that require shared resources. What companies might gain in savings, they lose in performance and flexibility. Not all applications have the same configuration needs.

From the application team's perspective, of course, this tradeoff is unacceptable. Developers are trying to build more dynamic, high-performing applications. If they cannot achieve this goal when faced with dynamic traffic loads, applications are essentially worthless. Yet when development teams ask for help from the infrastructure team in terms of configuring hardware ADCs for optimal performance, they are often met with resistance. The infrastructure team's priority is getting packets to a running machine; optimizing performance lies with the development team.

So, if not hardware, what's the answer? More modern ADC innovations, such as on-device virtualization and software-defined networking, might be a step up. However, even with extensive automation, these options are still inextricably linked to hardware or intensive manual configurations. They don't satisfy the impatient appetite of developers who just want to do their jobs—and not dance to the tune of the infrastructure team.

**Software-based ADCs, on the other hand, can tackle these challenges head-on and allow developers to do what they've been hired to do—create business value through innovative, engaging and high-performing applications**.

## Where does NGINX enter the picture?

When it comes to web application delivery, organizations have typically used separate components for the web server and the application delivery controller (ADC) or reverse proxy load balancer.  Often, the ADC was a hardware component—as we've mentioned previously—whose job was to aid in the performance of the web server to deliver online applications.

NGINX is changing that approach. Because creating a scalable and high-performance web server or ADC requires many of the same techniques, we've combined the two elements into a single software-based tool, essentially creating a new mechanism for web application delivery that offers performance and scalability at both the ADC and web server layers.

## Five Reasons to Switch to Software

Why does a software-based app delivery approach like NGINX surpass other options? Here are five reasons that will have you converting in no time.

If you've gone to the trouble to adopt virtual and/or cloud platforms for application development and delivery, software can help you maximize the benefit of those platforms in a way that hardware cannot.

Unlike legacy hardware ADCs, software ADCs are built natively to deploy anywhere. They slide right into cloud and virtual environments, with open APIs that enable integration with many of the software and virtualization tools you already use. The process is straightforward: download, configure, and test—with none of the risk associated with costly hardware purchases and complex configurations.

Similarly, because it is software-based, this kind of ADC approach provides much more flexibility when it comes to sizing and scaling for each application's needs. Whereas hardware limits you to expensive licenses for specific performance and throughput that's difficult to scale, software ADCs can be adjusted on demand when requirements change. This means developers have the freedom to deploy the right ADC for any application, quickly and easily. For example, you can deploy NGINX on the hardware of your choice, sized for your expected workload.  Alternatively, you can deploy multiple instances inside Linux containers and apply resource limits for full workload separation if necessary. Whether

## 1. You'll get the rapid deployment and flexibility you've been working toward.

deployed on hardware, cloud, or virtual platforms, NGINX offers fast deployment without compromise. This naturally provides a huge cost break as well. Software offers more flexible licensing options, so you never have to make rash predictions or buy more capacity than you need.

Remember, it’s an application-centric world, and websites are just one of many applications in a company’s full stack. NGINX enables developers to evaluate and deploy these apps faster, with lower cost and less risk.

## 2.

## You'll achieve more power and better performance.

The more apps that move to the cloud, the faster everything needs to move to keep pace. Picture Lucy and Ethel at the chocolate factory—it's an almost manic need for technology to scale upward, constantly, to support fluctuating and increasing computing velocity. This frenzy is symbiotically answering and driving user demand for nonstop availability and exceptional performance.  The cycle is relentless: Apps get more complex, in turn requiring more optimization, and traffic increases with business success, driving even more optimization, which further fuels performance expectations…and so on. Bottom line: Without better control, you cannot adequately ramp up performance. Only software ADCs like NGINX can meet those demands.

Hardware places artificial limits on load balancing that can be a nightmare to reset, whereas NGINX software will literally run as fast as you let it, whenever you want, and in any physical, virtual or cloud environment. The freedom from hardware configuration, licensing, and expenses means you can scale infrastructure—vertically or horizontally—on-demand, so every ADC is right-sized for the application it serves. You have the power to manage application delivery more effectively, so you can ultimately give applications the power they need in order to perform optimally.

## 3.

## You'll be able to run anywhere.

Your applications exist in real-time, and users are accessing them from an increasing variety of devices. This is one of the main reasons so many companies are moving to the cloud or virtual platforms—to be able to serve content quickly and dynamically to users on the go.

Legacy ADCs have a big problem here, however. Hardware can actually clog performance by tying your applications to specific physical locations, creating literal traffic jams. That's because hardware ADCs were built for a more traditional data center—one where applications weren't dynamic, where traffic flowed more predictably. Software load balancers, on the other hand, are built to run on today's more dynamic, virtualized architectures. In fact, NGINX can be dropped into any environment—whether on-premise, cloud, multi-cloud, or hybrid—offering load balancing that's portable and not tethered to one location or form factor. Your apps, in turn, have the support they need to provide incredible performance no matter where they are being served.

## 4.

## You'll finally be able to focus on your apps.

All applications are not alike, and neither are all client devices or users. That's why the confines of hardware infrastructures and configurations make little sense in an app-focused world. This is a world driven by applications, and those that cannot deliver exceptional performance will be cast aside by users. This is *reality*.

Yet when the cost and complexity of hardware ADCs force you to share resources across many applications, or pigeonhole applications into one-size-fits-all configurations, you are *compromising* performance. No wonder many application teams deem that unacceptable. If you force them to compromise on performance, you're tying their hands. To achieve peak performance for *every* app, developers need the freedom to make specific configurations on the load balancer without relying on the infrastructure team to make the changes. NGINX provides that freedom. Because it is software and can be rapidly configured by the application team, NGINX allows developers to designate a single load balancer per app, whenever needed, over and over again. As a result, application teams can focus on delivering optimal performance without distractions.

## 5. You'll have a DevOps-friendly solution.

We've mentioned that hardware ADCs and load balancers can create conflict between the application team and the infrastructure team. But replacing hardware with a software load balancer can eliminate that conflict by putting application performance management where it belongs: in the hands of the people who develop and operate the applications.

DevOps is the evolution of application development in a cloud-based and virtualized world. Application teams no longer build and then hand off to someone else to deploy; they must build, deploy, and repeat. Cloud environments have given developers the agility they need to craft and deliver continuously better applications and services—but hardware ADCs, or even virtualized ADCs tied to hardware, still kept them shackled.

When you un-tether the ADC from hardware, however, and take a software-based approach, you give control back to developers in a manner that's consistent with all your other software tools and licensing. This has significant implications on DevOps as a whole. Not only can applications be delivered with greater performance, but they can also now be developed to perform better *from the start*. Because it offers simple software configuration without high costs or complexity, application teams can use NGINX early in the development process to simulate production environments and test performance against specific traffic loads. App architects can better understand potential challenges and design more refined deployment policies that speak to specific application requirements, and as such, application delivery can be tackled as part of the development cycle—and not an afterthought.

Over the long run, this saves tremendous time, money, and effort—while resulting in the development of applications designed for optimal performance. The business as a whole benefits from this approach. The better applications perform, the more successful they become.

## Conclusion

As applications increase in number, complexity, and importance, the way in which we deliver applications must keep pace. Users expect continuously faster performance, and that's not likely to change. Hardware ADCs are one of the last lingering vestiges of an old data center architecture not designed to support this app-centric world. They obstruct performance, and they prevent developers from building and delivering applications that are optimized for performance.

By switching to a software load balancer or ADC, such as that provided by NGINX, you can cast off the limitations imposed by hardware to achieve newfound freedom and flexibility that extends throughout the business. From lower costs and less restrictive licensing to more scalability and improved development practices, NGINX helps drive the enterprise-class performance levels today's businesses—and users—demand.

For more information on how NGINX can help your company transition to software load balancing, contact us at http://nginx.com/contact/. Or request your free trial today: http://nginx.com/free-trial-nginx-plus/

Nginx, Inc. offers advanced Internet infrastructure software that enables companies to match increasing demand for faster web experiences at scale. Nginx offers two solutions: NGINX is the popular open source software powering much of the world's largest and fastest websites; NGINX+ is a commercial offering that includes all the power and scale of NGINX, plus additional, enterprise-class features such as application balancing, health checks, activity monitoring, on-the-fly configuration, and support.