SCALABILITY RULES

50 PRINCIPLES FOR SCALING WEB SITES

MARTIN L. ABBOTT MICHAEL T. FISHER

"Whether you're taking on a role as a technology leader in a new company or you simply want to make great technology decisions, Scalability Rules will be the go-to resource on your bookshelf."
— **Chad Dickerson,** CTO, Etsy

Praise for *Scalability Rules*

"Once again, Abbott and Fisher provide a book that I'll be giving to our engineers. It's an essential read for anyone dealing with scaling an online business."

—**Chris Lalonde**, VP, Technical Operations and Infrastructure Architecture, Bullhorn

"Abbott and Fisher again tackle the difficult problem of scalability in their unique and practical manner. Distilling the challenges of operating a fast-growing presence on the Internet into 50 easy-to-understand rules, the authors provide a modern cookbook of scalability recipes that guide the reader through the difficulties of fast growth."

—**Geoffrey Weber**, Vice President, Internet Operations, Shutterfly

"Abbott and Fisher have distilled years of wisdom into a set of cogent principles to avoid many nonobvious mistakes."

—**Jonathan Heiliger**, VP, Technical Operations, Facebook

"In *The Art of Scalability*, the AKF team taught us that scale is not just a technology challenge. Scale is obtained only through a combination of people, process, *and* technology. With *Scalability Rules*, Martin Abbott and Michael Fisher fill our scalability toolbox with easily implemented and time-tested rules that once applied will enable massive scale."

—**Jerome Labat**, VP, Product Development IT, Intuit

"When I joined Etsy, I partnered with Mike and Marty to hit the ground running in my new role, and it was one of the best investments of time I have made in my career. The indispensable advice from my experience working with Mike and Marty is fully captured here in this book. Whether you're taking on a role as a technology leader in a new company or you simply want to make great technology decisions, *Scalability Rules* will be the go-to resource on your bookshelf."

—**Chad Dickerson**, CTO, Etsy

consider that they hold very little value to a potential bad guy. Given the high cost in terms of potential availability impact and capital relative to the likelihood that they are the focus of a bad guy's intentions, it makes little sense for us to invest in their protection. We'll just ensure that they are on private IP space (for example, 10.X.Y.Z addresses or the like) and that the only traffic that gets to them are requests for ports 80 and 443.

On the flip side we have items like credit cards, bank account information, and social security numbers. These items have a high perceived value to our bad guy. They are also less costly to protect relative to other objects as they tend to be requested less frequently than many of our objects. We absolutely should lock these things away!

In the middle are all the other requests that we service within our platform. It probably doesn't make a lot of sense to ensure that every search a user performs goes through a firewall. What are we protecting? The actual servers themselves? We can protect our assets well against attacks such as distributed denial of service attacks with packet filters, routers, and carrier relationships. Other compromises can be thwarted by limiting the ports that access these systems. If there isn't a huge motivation for a bad guy to go after the services, let's not spend a lot of money and decrease our availability by pretending that they are the crown jewels.

In summation, don't assume that everything deserves the same level of protection. The decision to employ firewalls is a business decision focused on decreasing risk at the cost of decreasing availability and increasing capital costs. Too many companies view firewalls as a unary decision—if it exists within our site it must be firewalled when in fact firewalls are just one of many tools you might employ to help decrease your risk. Not everything in your product is likely deserving of the cost and impact to availability that a firewall represents. As with any other business decision, this one should be considered in the light of these tradeoffs, rather than just assuming a cookie-cutter approach to your implementation. Given the nature of firewalls, they can easily become the biggest bottleneck from a scale perspective for your product.

Rule 16—Actively Use Log Files

> **Rule 16: What, When, How, and Why**
>
> **What:** Use your application's log files to diagnose and prevent problems.
>
> **When to use:** Put a process in place that monitors log files and forces people to take action on issues identified.
>
> **How to use:** Use any number of monitoring tools from custom scripts to Splunk to watch your application logs for errors. Export these and assign resources for identifying and solving the issue.
>
> **Why:** The log files are excellent sources of information about how your application is performing for your users; don't throw this resource away without using it.
>
> **Key takeaways:** Make good use of your log files, and you will have fewer production issues with your system.

In the spirit of using the right tools for the job, one of the tools that is likely in all our toolboxes but often gets overlooked are log files. Unless you've purposely turned off logging on your Web or application servers almost all varieties come with error and access logs. Apache has error and access logs, Tomcat has java.util.logging or Log4j logs, and Websphere has SystemErr and SystemOut logs. These logs can be incredibly valuable tools for providing insights into the performance and problems occurring within your application that might prevent it from scaling. To best use this tool there are a few simple but important steps to follow.

The first step in using log files is to aggregate them. As you probably have dozens or perhaps even hundreds of servers, you need to pull this data together to use it. If the amount of data is too large to pull together there are strategies such as sampling, pulling data from every n^{th} server, which can be implemented. Another strategy is to aggregate the logs from a few servers onto a log server that can then transmit the semi-aggregated logs into the final aggregation location. As shown in Figure 4.4, dedicated log servers can aggregate the log data to then be sent to a data store. This aggregation is generally done through an out-of-band network that is not the same network used for production

traffic. What we want to avoid is impacting production traffic from logging, monitoring, or aggregating data.

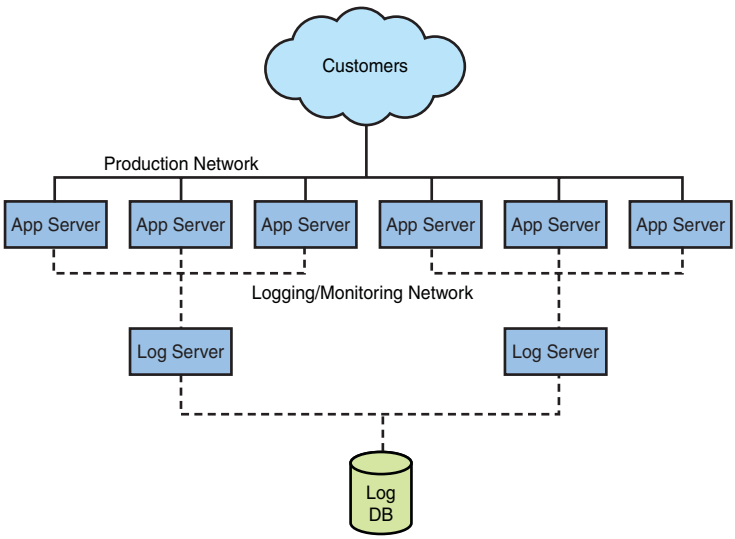

Figure 4.4 Log aggregation

The next step is to monitor these logs. Surprisingly many companies spend the time and computational resources to log and aggregate but then ignore the data. While you can just use log files during incidents to help restore service, this isn't optimal. A preferred use is to monitor these files with automated tools. This monitoring can be done through custom scripts such as a simple shell script that greps the files, counting errors and alerting when a threshold is exceeded. More sophisticated tools such as Cricket or Cacti include graphing capabilities. A tool that combines the aggregation and monitoring of log files is Splunk.

Once you've aggregated the logs and monitored them for errors, the last step is to take action to fix the problems. This requires assigning engineering and QA resources to identify common errors as being related to individual problems. It is often the case that one bug in an application flow can result in many different error manifestations. The same engineers who

identified the bug might also get assigned to fix it, or other engineers might get assigned the task.

We'd like to have the log files completely free of errors, but we know that's not always possible. While it's not uncommon to have some errors in application log files you should establish a process that doesn't allow them to get out of control or ignored. Some teams periodically, every third or fourth release, clean up all miscellaneous errors that don't require immediate actions. These errors might be something as simple as missing redirect configurations or not handling known error conditions in the application.

We must also remember that logging comes at some cost. Not only is there a cost in keeping additional data, but very often there is a cost in terms of transaction response times. We can help mitigate the former by summarizing logs over time and archiving and purging them as their value decreases (see Rule 47). We can help minimize the former by logging in an asynchronous fashion. Ultimately we must pay attention to our costs for logging and make a cost-effective decision of both how much to log and how much data to keep.

Hopefully we've convinced you that log files are an important tool in your arsenal of debugging and monitoring your application. By simply using a tool that you likely already have, you can greatly improve your customer experience and scalability of your application.

Summary

Using the right tool for the job is important in any discipline. Just as you wouldn't want your plumber to bring only a hammer into your house to fix your pipes, your customers and investors don't want you to bring a single tool to solve problems with diverse characteristics and requirements. Avoid falling prey to Maslow's hammer and bring together diverse teams capable of thinking of different solutions to problems. A final word of caution on this topic is that each new technology introduced requires another skill set to support. While the right tool for the job is important, don't overspecialize to the point that you have no depth of skills to support your systems.

Endnotes

1. Jeffrey Dean and Sanjay Ghernawat, “Map Reduce: Simplified Data Processing on Large Clusters,” Google Research Publications, http://labs.google.com/papers/mapreduce.html.

This page intentionally left blank