6 Ways to Kill Your Servers - Learning How to Scale the Hard Way

Monday, August 23, 2010 at 7:58AM

Todd Hoff in Strategy

This is a guest post by Steffen Konerow, author of the High Performance Blog.

Learning how to scale isn't easy without any prior experience. Nowadays you have plenty of websites like highscalability.com to get some inspiration, but unfortunately there is no solution that fits all websites and needs. You still have to think on your own to find a concept that works for your requirements. So did I.

A few years ago, my bosses came to me and said "We've got a new project for you. It's the relaunch of a website that has already 1 million users a month. You have to build the website and make sure we'll be able to grow afterwards". I was already an experienced coder, but not in these dimensions, so I had to start learning how to scale – the hard way.

The software behind the website was a PHP content management system, based on Smarty and MySQL. The first task was finding a proper hosting company who had the experience and would also manage the servers for us. After some research we found one, told them our requirements and ordered the suggested setup:

- LoadBalancer (+Fallback)
- 2 Webservers

- Mysql Server (+Fallback)
- development machine

They said, that’s gonna be all we need – and we believed it. What we got was:

- Loadbalancer (single core, 1GB RAM, Pound)
- 2 Webservers (Dual core, 4GB RAM, Apache)
- MySQL Server (Quad core, 8GB RAM)
- Dev (single core, 1GB RAM)

The setup was very basic without any further optimization. To synchronize the files (php+media files) they installed DRBD in active-active configuration.

Eventually the relaunch came – of course we were all excited. Very early in the morning we switched the domains to the new IPs, started our monitoring scripts and stared at the screens. We had almost instantly traffic on the machines and everything seemed to work pretty fine. The pages loaded quickly, MySQL was serving lots of queries and we were all happy.

Then, suddenly our telephones started to ring “We can’t access our website, what’s going on?”. We looked at our monitoring software and indeed – the servers were frozen and the site offline! Of course, the first thing we did was calling our hoster “hey, all our servers are dead. What’s going on?”. They promised to check the machines and call back immediately afterwards. The call came: “well,…erm… your filesystem is completely fubar. What did you do? It’s totally screwed”. They stopped the loadbalancers and told me have a look at one of the webservers. Looking at the index.php file I was shocked. It contained some weird fragments of C code, error messages and something that looked like log

files. After some further investigation we found out that DRBD was the cause for this mess.

Lesson #1 learned

Put Smarty compile and template caches on an active-active DRBD cluster with high load and your servers will DIE!

While our hoster was fixing the webservers I rewrote some parts of the CMS to store the Smarty cache files on the servers local filesystems. Issue found & fixed. We went back online! Hurray!!!

Now it was early afternoon. The website usually reaches its peak in the late afternoon until early evening. At night the traffic goes back to almost none. We kept staring at the monitoring software and we were all sweating. The website was loading but the later it got, the higher the system load and the slower the responses. I increased the lifetime of the Smarty template caches and hoped it would do the trick – it didn't! Very soon the servers started to give timeouts, white pages and error messages. The two machines couldn't handle the load.

Our customer got a bit nervous at the same time, but he said: Ok, relaunches usually cause some issues. As long as you fix it quickly, it will be fine!

We needed a plan to reduce the load and discussed the issue with our hoster. On of their administrators came up with a good idea: "Guys, your servers are currently running on a pretty common Apache+mod_php setup. How about switching to an alternative webserver like Lighttpd? It's a fairly small project, but even wikipedia is using it". We agreed.

Lesson #2 learned

Put an out-of-the-box webserver configuration on your machines, do not optimize it at all and your servers will DIE!

The administrator gave his best and reconfigured both webservers as quickly as he could. He threw away the Apache configuration and switched to Lighttpd+FastCGI+Xcache. Later, when we went back online we almost couldn't stand the pressure anymore. How long will the servers last this time?
The servers did surprisingly well. The load was MUCH lower than before and the average response time was good. After this huge relief we went home and got some sleep. It was already late and we came to the conclusion there was nothing left we could do.

The next days the website was doing rather well, but at peak times it was still close to crash. We spotted MySQL as the bottleneck and called our hoster again. They suggested a MySQL Master-Slave replication with a MySQL slave on each webserver.

Lesson #3 learned

Even a powerful database server has its limits and when you reach them – your servers will DIE!

In this case the database became so slow at some point, that the incoming and queued network connections killed our webservers – again. Unfortunately this issue wasn't easy to fix. The content management system was pretty simple in this regard and there was no built-in support for separating reading and writing SQL queries. It took a while to rewrite everything, but the result was astonishing and worth every minute of suspended sleep.

The MySQL replication really did the trick and the website was finally stable! YEAH! Over the next weeks and months the website became a

success and the number of users started to increase constantly. It was only a matter of time until the traffic would exceed our resources again.

Lesson #4 learned

Stop planning in advance and your servers are likely to DIE.

Fortunately we kept thinking and planning. We optimized the code, reduced the number of needed SQL queries per pageload and suddenly stumbled upon MemCached. At first I added MemCached support in some of the core functions, as well as in the most heavy (slow) functions. When we deployed the changes we couldn't believe the results – it felt a bit like finding the Holy Grail. We reduced the number of queries per second by at least 50%. Instead of buying another webserver we decided to make even more use of MemCached.

Lesson #5 learned

Forget about caching and you will either waste a lot of money on hardware or your servers will die!

It turned out, that MemCached helped us to reduce the load on the MySQL servers by 70-80%, which also resulted in a huge performance boost – also on the webservers. The pages were loading much quicker!

Eventually our setup seemed to be perfect. Even on peak times we didn't have to worry about crashes or slow responding pages anymore. Did we make it? No! Out of the blue one of the webservers started having some kinda hickups. Error messages, white pages and so on. The system load was fine and in most cases the server worked, but only in "most cases".

Lesson #6 learned

Put a few hundred thousand small files in one folder, run out of Inodes and your server will die!

Yes you read it correct. We were so focused on MySQL, PHP and the webservers itself that we didn't pay enough attention to the filesystem. The Smarty cache file were stored on the local filesystem – all in one single directory. The solution here was putting Smarty on a dedicated ReiserFS partition. Furthermore we enabled the Smarty "use_subdirs" option.

Over the past years we kept optimizing the pages. We put the Smarty caches into memcached, installed Varnish to reduce the I/O load for serving static files more quickly, switched to Nginx (Lighttpd randomly produced error 500 messages), installed more RAM, bought better hardware, more hardware… this list is endless.

Conclusion

Scaling a website is a never ending process. As soon as you fix one bottleneck you'r very likely to stumble into the next one. Never ever start thinking "that's it, we're done" and lean back. It will kill your servers and perhaps even your business. It's a constant process of planning and learning. If you can't get a job done on your own, because you have a lack of experience and/or resources – find a competent and realiable partner to work with. Never stop talking with your team and partners about the current issues and the ones that might arise in (near) future. Think ahead and be proactive!
