Fun with SMTP Relay

Posted At : December 6, 2011 2:52 PM | Posted By : Mark Kruger
Related Categories: Hosting and Networking

This is a post about solving a particular problem with SMTP relay that involves mass emails. Whenever I write a post on this topic there are 2 things that my savvy readers always feel compelled to tell me:

- **"Hey Muse, make sure you are not sending SPAM"** - Thanks for the advice. The Muse takes a dim view of SPAM. Like everyone else I'm tired of being told I've won the lottery, have friends in Nigeria, and need to be more concerned about my size. The emails in this case are *not* spam - but if you are tempted to make that comment I appreciate that you have the Muse' back.
- **"Hey Muse, you are crazy to do this yourself** - Thanks for that as well. And please don't hesitate to tell me (again) about the various services that are out there - all of which are better equipped technically, mentally, physically, and ecumenically to handle my email so that I need not be an expert on the topic. I always appreciate that input. The only thing I like better than leaving money on the table is transferring it to another vendor. In fact it's an axiom of business to never do anything for money that you can pay someone else to do for you. Next week I'm writing about server troubleshooting and that will give you an additional opportunity to tell me (again) about how that no one needs to worry about that anymore either because the cloud fairies do it all magically.

Now that the preliminary caveats are out of the way, let's talk environment, then problem then solution.

The Setup

This particular customer sends around 500,000 marketing emails (coupons, specials and advertisements to existing customers who are eager to get them) in a single send a few times a week. So let's say 2 to 3 million emails per week. During the holidays this number is multiplied by 5 to 10 to 15 million emails. This mailing list has grown over the years as well. The number of sends that go out now are double what they were a few years ago.

To accomplish a "mailer" (a send of a group of 500,000 emails), the messages are queued in a table and then sent in groups of a few thousand at a time till the queue table is empty. Sends are done using ColdFusion 8 enterprise with a scheduled task. The system runs on 2 load balanced servers with Microsoft IIS SMTP as the relay agent. It has been in service for about 2-3 years with no performance issue other than a general uptick in processor and memory usage during a send.

The Problem

In early November of this year we launched a new version of the site. It's a stunning new version with more functionality, a beautiful look, great browsing and ordering capabilities etc. The result was an uptick in traffic commensurate with the email sends. Whereas before we would see processor and memory usage jump during a send, now that same jump was resulting in general slowness. At times we would have to restart CF on one of the servers to "kick start" its speed. During those times however, CF did not seem to be under any unusual duress - besides the increased request handling which we had planned for. To put it another way, our mailers were not really

using more hardware resources than usual, but they definitely seemed to causing a lag.

Indeed, a closer look found that it was solely the mail process causing the issue. It wasn't necessary to restart CF at all. Simply restarting the SMTP service resolved the speed issue without doing anything to IIS Web or to ColdFusion. And the server stayed performing well until the email queue got another 80-90k messages in it. So at this slightly enhanced resource level the IIS SMTP service could not keep up. Looking at the process viewer I could see the "inetinfo.exe" process - which normally runs with a very small memory footprint of 10 to 40 megs - was now running with 300 to 400 megs of memory. In fact, in spite of our experience with sending millions of emails through it, IIS SMTP is simply not designed to do mass email very well. It is a fairly crude service with very little that you are able to configure. IIS SMTP runs inside of the inetInfo.exe process - which is doing double duty as a controller for resources allocated to web sites *and* as an SMTP service delivering mail. It's simply ill-suited for double-duty, although it handles a random trickle "trickle" email just fine.

Our first troubleshooting attempts focused on fine-tuning either the send process or IIS itself. But none of the settings in the SMTP setup for IIS had an appreciable enough effect to make a difference. Part of the reason is greylisting and other "delay" based spam mitigation techniques mean that delivering these messages is not a "smooth flow". It's more like getting the last one third of ketchup from the bottle. The first two thirds go fine but that last third requires a lot of shaking and pounding.

And this brings up another problem - a problem that had actually bedeviled this site from the beginning (the site owners had simply learned to live with it). It has to do with email stacking up. During a mailer, as many as 120,000 messages might build up in the IIS SMTP queue waiting for initial delivery or retries. Now we don't care (much) about the length of time it takes to deliver such messages. After all if the first marketing message goes out right away but the last one has a cup of coffee and isn't delivered for 30 or 40 minutes - so what. The recipient didn't know it was coming and doesn't know what he or she is missing right?

But remember that marketing messages were *not the only ones going into the queue*. SMTP also had to support the typical "trickle traffic" of a busy ecommerce site. Unlike our marketing emails these emails are quite time sensitive. They include order and shipping notifications, customer service requests, wish lists, password recovery requests etc. - all the activity that a functioning site needs to service its customers. These more important emails would often be placed at the bottom of the stack - forced to wait for thousands of marketing emails to "clear out" before they were delivered. Our customer had been living with these delays during a mailer for some time. They simply knew that during a mailer these emails would sometimes be delayed significantly.

The Fix

Our proposal was to find a solution that would allow us to relay the marketing email to a secondary relay service. This would fix our performance problem and also allow our "trickle" email to be delivered in a timely manner. Our task was to find a solution that allowed for the following:

- It could be used with existing hardware - no new server would be necessary.
- Fairly cost effective - not looking for a full-featured email server after all.
- Because we wanted to make as few changes as possible we needed it to be on the

same "local" server but on a secondary port.
- Allowed us to "fine tune" relaying. Whitelisting, specific domain impersonation, smart gateway functionality etc.
- Operated without conflict with existing IIS SMTP service which would still be used as the relay for time sensitive emails.
- Enabled me to "turn off" features and run *only* those necessary for SMTP relay.

A quick note to my friends on the guru list teased out several recommendations. I finally settled on one called **Mail Enable** - a .NET application with lots of tuning options. For a single domain the free version looked like it would work quite nicely.

Of course before I proceeded I tested on dev server running IIS with SMTP delivery. The only hiccup I ran into was that I had to enable IIS 6 compatibility mode (using the features module). Other than that the install was seamless. During the install it warned me that another SMTP server was running and allowed me to pick a separate port. It also allowed me to *not* install the web based mail (which would have been added to my sites - something I clearly don't need or want).

Once installed it gave me fine tuned control over incoming and outgoing connections - enabling me to specify the level of resources that I wanted this secondary relay to consume. I tested both relays by sending mail to external email accounts through both services (IIS SMTP on port 25 and mailenable SMTP on my new custom port 2525) and verified that emails from both services were delivered. Then I tried queuing up a few thousand emails on my new mail server and while they were outgoing I sent a few through IIS SMTP - to ensure that trickle email was going out in real time as the marketing email was also being delivered - so far so good. Finally I sent a few emails through both services and examined the headers. This let me fine tune the headers and host entries to ensure my best chance at delivery. Of course I had to make some code changes to my mailer process to point to the new server and port - but they were trivial changes.

Once in place I had a way of managing a controlled spigot of marketing email that did not interfere with IIS and did not delay regular trickle email any longer. The end result was faster delivery over all from a better equipped relay server. To my pleasant surprise *Mail Enable* ran very efficiently - spooling up and down the memory footprint and thread and handle count as needed. When idle it used about 6 or 7 threads and less than 10 megs. It seems a well-designed application and very stable. The server side admin interface is a little counter intuitive - but I got the hang of it after 10 minutes or so.

Conclusion

Remember - I'm aware there are many ways to solve this sort of problem. I also realize if I was relaying through a service I wouldn't have this issue. But as always the Muse welcomes your comments and tips.