Stress Testing a CF Server - Interesting results

Posted At : November 24, 2008 6:47 PM | Posted By : Tariq Ahmed
Related Categories: Tools of the Trade

Over the years we've been enhancing this fairly large (500K+ lines) CF application. It's a system that we've inherited, and as we've built up the infrastructure we generally knew what the operating capacity needed was.

As part of a Disaster Recovery project we've relocated the environment to a state of the art Colo, so as part of building a brand new home for this product we wanted to do some stress testing. Part of that was to observe how the Web/CF/DB servers perform, but this is the first time users would be accessing the servers over a WAN - so we wanted to simulate what the experience will be like when the WAN has heavy HTTP load on it.

Tool Selection

We needed to quickly find some tools to do this, so we took a look at a few products including Microsoft's Web Application Stress Tool, Minq's PureLoad, Paessler's Webserver Stress Tool 7, and Apache's jMeter.

Microsoft's tool seems to have technical potential, but from a usability point of view it's difficult to use. PureLoad is industrial strength, and a great solution if you need to do a lot of it regularly, but the setup is too involved for a quick and simple test. Similarly with jMeter, it's an industrial stength tool, but there's no way non-technical people would be able to quickly get it setup.

So that left us with Paessler's WebServer Stress Tool (PWST); it's got great features and the ability to do complex scripting. But our use case was to have a bunch of people load it up on their desktops and fire away with a barrage of canned URLs. And PWST can help you do that in a matter of minutes.

Granted, when the URLs remain static you'll start to get caching going on at very levels of the system, but we made sure everyone was generating their own unique set of URLs that even if cached were returning significant amounts of data.

Observing the Results

Our environment consists of a CISCO Network Load Balancer using round robin w/sticky sessions to four ColdFusion 8 Enterprise servers.

We progressively ramped up more and more load and could see the load being distributed fairly evenly, and CPU and Memory increased at a linear rate.

Another goal was to progressively increase the load towards a point of failure, and as the load ramped up towards **10X more load** than what the servers would ever see, it was interesting to see how the system failed. I was expecting for one server to lock up first, and then the others to pick up the slack but act incredibly sluggish before another went down.

What actually happened is within 10-15 seconds of the first server going down, the load was re-distributed and the remaining servers were already on the brink of failure that they all went down pretty much at the same time.