Mike Brunt's Pre-CFUnited Blog

Posted At : June 9, 2008 11:40 AM | Posted By : Matt Weiss
Related Categories: Test

I will be presenting at CFUnited on Friday June 20, 2008, on the subject of High Availability (HA)-Clustering for ColdFusion/JRun applications and I intend to make this as practical as possible. Having spent many years travelling the world helping to fix slow or unresponsive ColdFusion applications, I see HA as a natural progression to this and in fact Load-Balancing, which is a part of Clustering, has a direct impact on improving performance.

There is a point, often overlooked by even the manufacturers of clustering/load balancing equipment. Clustering is the overall term which, in my opinion, applies whenever two items or more appear as one, to the users. In our world, that typically means multiple web servers, with multiple application and database servers.

With this aspect of Clustering there are two services which are a part of the Clustering; **Fail-Over** and **Load-Balancing**. In my experience **Fail-Over** is always present, meaning if one member of the Cluster fails the remaining members ensure that continuity of service is maintained. This is a prime function of a Cluster.

Load-Balancing is the apportioning of load around members of the Cluster, typically an even distribution of the load is what is required. The most even distribution is via Round-Robin which means each single request moves around the Cluster members, like this (this example shows a 3 member Cluster):

- USER 1 > REQUEST1 > CLUSTERMEMBER1
- USER 2 > REQUEST1 > CLUSTERMEMBER1
- USER 1 > REQUEST2 > CLUSTERMEMBER2
- USER 2 > REQUEST2 > CLUSTERMEMBER2
- USER 1 > REQUEST3> CLUSTERMEMBER3
- USER 2 > REQUEST3> CLUSTERMEMBER3
- USER 1 > REQUEST4 > CLUSTERMEMBER1
- USER 2 > REQUEST4 > CLUSTERMEMBER1

This is the most evenly balanced Load Balancing algorithm and as I mentioned above is the Round-Robin algorithm. Problems can occur with that algorithm if there are user specific items in memory on one of the Cluster members, for instance in memory session state variables. If USER1 has logged in to CLUSTERMEMBER1 above and their details are in session variables on CLUSTERMEMBER1 when users next request takes them to CLUSTERMEMBER2 those in memory session state variables will not be there. My preference for the optimal Load-Balancing algorithm is Round-Robin with Sticky Sessions. In the case a user “sticks” to a Cluster member as follows:

- USER 1 > REQUEST1 > CLUSTERMEMBER1
- USER 2 > REQUEST1 > CLUSTERMEMBER2

- USER 1 > REQUEST2 > CLUSTERMEMBER1
- USER 2 > REQUEST2 > CLUSTERMEMBER2
- USER 1 > REQUEST3> CLUSTERMEMBER1
- USER 2 > REQUEST3> CLUSTERMEMBER2
- USER 1 > REQUEST4 > CLUSTERMEMBER1
- USER 2 > REQUEST4 > CLUSTERMEMBER2

This is not quite as evenly balanced as Round-Robin alone but unless there is a failure of one Cluster member the user will not lose their session state variables and the load balances across all Cluster members eventually.

This article serves as the first in a series of posts leading up to the CFUnited presentation and my next one will delve into differences between Hardware and Software Clustering.