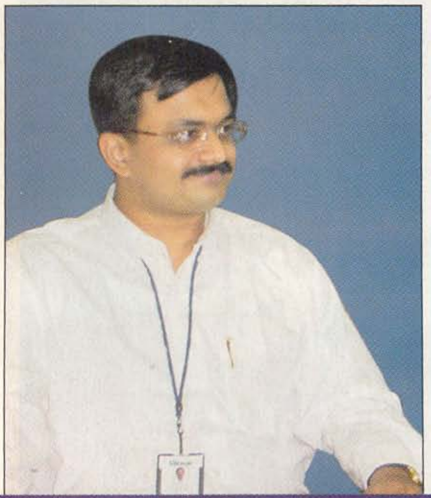
The biggest tech challenge I faced and how I overcame it

TNC Venkata Rangan
CMD, Vishwak Solutions and Regional Director, MSDN

is passionate about architecting mobile and Web solutions around .NET technologies. You can contact him via vishwak.com



Challenge

Background

It was about a year back that a leading global portal approached us. The company had over a dozen websites serving about six countries in Asia alone. They wanted a solution that could track the number of times each advertising image was served. They also wanted secure, customized report for each advertiser. We had to deploy the solution in their existing web-farm, which had a couple of Web servers (Win 2000 Server, IIS) and one database server (MS SQL Server 2000).

Challenges

The websites were heavily visited, and the client wanted us to serve and track over 2 million images a day. The image-tracking system was to be such that it should not have missed even a single image served and reporting had to be real time or near-real time. The specification did not allow us to add more hardware.

Options

1) Updating the back-end database every time an image was served (using ADO and ASP). We had to rule out this option as doing 2 million database connections in a day with just one database server is impossible.
2) Maintaining an in-memory cache in each Web server and periodically updating the backend database. This option had to be ruled out has we did not have the time to design, build, test and deliver a reliable caching system. Moreover, an in-memory cache would be lost in the eventuality of a server restart or network breakdowns.

Eureka!

MSMQ (Microsoft Message Queue Server) came to our rescue. Out-of-the box MSMQ provided a reliable, asynchronous queuing mechanism. We designed a system with the master queue in the database server and the independent slave queues on each of the Web servers. Every time an image was served, we wrote a message locally to the slave queue and MSMQ ensured that this message got moved to the master queue. MSMQ provides guaranteed message delivery, which made our solution resilient. A VB.NET service, running in the database server, periodically read the messages from the master queue, opened a connection and updated the database. As the service ran every 5 mins (configurable), we could do about 7000 message updates (equal to 7000 images served) with a single DB connection. This solution could also scale up by re-configuring the time gap between each service run.

Learnings

Using a ready-made wheel is better than reinventing one! Using out-of-the box tools reduces delivery time and increases the reliability of solutions.