Jon's Performance Musings: Ah, Yes, Customers!

Providing IT services would be really easy if one didn't have to deal with customers. We could take as much time as we needed to develop the application, debug it, and tune it. Without customers there wouldn't be demand on the system, and there probably wouldn't be any performance problems. On the other hand, we would be impoverished, since there wouldn't be a revenue stream either.

Customers are a necessary evil.

So customers are a necessary evil. It's a happy problem to have, since the more customers there are, theoretically the more revenue there is, and we can all draw our salaries. On the other hand, customers can create problems, either individually or in the aggregate. Here are a couple of tales to amuse and elucidate.

The number-cruncher.

A real estate database company was concerned about the performance of their system. They had been having great performance, but over the last several weeks the system was slowing down. Worse, they were seeing very high utilization across all of their CPUs. They didn't want to upgrade unless they were forced into doing so, but they were very concerned with the quality of service that they were providing their clients.

Flat line.

The odd thing about the usage of their system is that the CPUs were always busy. Twenty-four hours a day, seven days a week. Normally, in an interactive system where live customers are creating demand, the demand varies by time of day. People arrive at their desks in the morning, break for lunch, and go home at night. This creates a non-constant pattern of demand on the system. In most cases, usage of the system follows that pattern. That didn't appear to be happening on this system. Why?

No logs.

The challenge here is that the application didn't log the demand. Apparently their customers paid a flat fee to access the system as much as they wanted. There was no attempt to record who accessed or how often the clients accessed the system, nor to find out what or how much data they accessed. Frankly, that is a poor way to design an application, but that's what they chose to do. In our experience customers will always do the unexpected, and an appropriate level of logging will help to point out what's really going on.

Dig.

With classic black-box analysis we were able to identify the processes that were using CPU. From this the client could identify which application was being used. We could also tell that the SQL process was mostly reading the database, with only a few inserts. We began to suspect

that someone or something was submitting a constant stream of queries against the database. We could also identify which IP processes were involved. We were able to identify the IP address which was sending a lot of data outbound, and since customers arrived via different IP addresses, we could identify a subset of customers who might be causing the traffic.

Beware of customers bearing scripts.

Armed with this information, the company was able to identify a suspect "number cruncher" who was known to have a fleet of PCs which were submitting scheduled scripts with heavy queries. We understand that final resolution was to give that person a new IP address, and see if the problem moved. It did.

Perfect scripting? Not!

Hard to believe (!) but occasionally customer scripts are less than perfect. Another company has a world-wide network of systems doing payment authorization and settlement. One of their customers had a script fail, with the result that it continually submitted the same file via FTP across their network. That cannot have been good for overall network performance, even though the application rejected the resubmission. Performance statistics and logs pinpointed the problem.

Beware success.

Systems that process retail point of sale transactions have particular challenges. The frenzy that occurs during the holiday season can strain a well-prepared system, especially if something occurs to bust the predictions. Depending on the organization, the holiday retail POS peak can be from 50% to 500% of the usual non-holiday peak. Most organizations recognize this, and plan for it.

Know the business.

The experience that this particular organization had was a result of success. It was caused by a failure of human communication, and/or an under-estimate of how successful the product would be. This retailer added a new gift-card requiring a new link to an authorizer. During the lead-up to Christmas the link worked fine, and lots of gift cards were sold. However, no one thought about the fact that in the first couple of days after Christmas all of those cards would come back to be redeemed. They did, the link to the back-end was overwhelmed, and lots of customers and staff were impacted.

Know the customer.

These experiences contain a couple of lessons. First, give yourself the tools to understand what the customer is doing. Get a handle on their pattern of usage, and learn to anticipate what's coming. Use the tools to investigate issues, and plan to eliminate their cause. Second, always talk to your business partner. Our systems aren't there just for our enjoyment and employment. They and we are there to service our customers.

Jon E. Schmidt
Transaction Design, Inc.
San Rafael, CA, 94901, USA
1.415.256.8369
inform@banbottlenecks.com

Jon is the founder of Transaction Design, Inc. (TDI), a consulting firm located in the San Francisco Area which specializes in quality of service studies with clients worldwide. He is the creator of the Ban Bottlenecks® service and has an extensive background in the implementation, testing, and tuning of high-availability systems.