Take it easy on the automatic retries

devblogs.microsoft.com/oldnewthing/20051107-20

November 7, 2005

Raymond Chen

When I saw a discussion of how to simulate retry via try/catch, using as inspiration a Ruby function that retried a network operation three times before finally giving up, I felt the need to caution against automatic retry.

Your natural inclination when faced with a failure that has a good chance of being caused by a transient condition is to retry it a few times. The second, or possibly third, try will finally work, and your function can continue. The user gets what they want without an annoying "Abort, Retry, Cancel"-type dialog, one less support call for you. What could possibly go wrong?

I've seen this go wrong many times. So much so that my personal recommendation is simply never to retry automatically. If something fails, then report the failure. If the user wants to retry, let them be the ones to make that decision.

Here's how it goes wrong. This is a real example, but the names have been removed because I'm not trying to ridicule anybody; I want you to learn. There was a networking feature that implemented some type of distributed networking capability. It is the nature of networks to be unreliable, so the implementors of the functionality decided to retry ten times before finally giving up. The operation they were performing was implemented by another group, and that other group also decided to retry five times before giving up. That second group called a networking function with a timeout of thirty seconds. Meanwhile, the application that used this networking capability attempted the operation fifteen times.

Let's do some math. At the bottom was a timeout of thirty seconds. Five retries comes out to two and a half minutes. Ten retries from the next layer brings this to twenty-five minutes. Fifteen retries from the application layer takes us to over six hours. An operation that would normally have completed (with a failure code) in thirty seconds became, through the multiplicative effect of multiple layers of retrying, a six-hour marathon. And then you get a very angry call from one of your customers demanding that you deliver them a fix yesterday because this problem is taking down their entire sales force.

(Explorer is hardly blameless in this respect. Though the article's attempt to patch shell32.dll is doomed to failure since shell32.dll is frequently updated by security patches.)

Raymond Chen

Follow