Scrutinize resource pools

A cascading failure often results from a resource pool, such as a connection pool, that gets exhausted when none of its calls return. The threads that get the connections block forever; all other threads get blocked waiting for connections. Safe resource pools always limit the time a thread can wait to check out a resource.

Defend with Timeouts and Circuit Breaker

A cascading failure happens *after* something else has already gone wrong. Circuit Breaker protects your system by avoiding calls out to the troubled integration point. Using Timeouts ensures that you can come back from a call out to the troubled one.

4.4 Users

Users are a terrible thing. Systems would be infinitely more stable without them. The human users of a system have this knack for creative destruction. When your system is teetering on the brink of disaster like a car on a cliff in a movie, some user will be the seagull landing on the hood. Down she goes! Human users have a gift for doing exactly the worst possible thing at the worst possible time.

Users are a terrible thing.

Worse yet, other systems that call ours march remorselessly forward like an army of Terminators, utterly unsympathetic about how close we are to crashing.

Traffic

Every user consumes some system resources. Unless you are building a peer-to-peer system such as BitTorrent, your system's capacity is limited. It scales with the amount of hardware and bandwidth you've bought, not the number of users you've attracted.

As traffic grows, it will eventually surpass your capacity.[8] Then comes the biggest question: How does your system react to excessive demand?

In Section 8.1, *Defining Capacity*, on page 135, we will see the definition of capacity: when transactions take too long to execute, it means that the demand on your system has exceeded its capacity. Internally to your system, however, there are some harder limits. Passing those limits makes cracks in the system, and cracks always propagate faster under stress.

8. If traffic isn't growing, then you have other problems to worry about!