Cold, warm, and hot standby

So far, we've said that to make an HA system, we need to have some way of restarting failed components. But we haven't discussed how, or what impact it has.

Recall that when a failure happens, we've just blown the MTBF number — regardless of what the MTBF number is, we now need to focus on minimizing the MTTR. Repairing a component, in this case, simply means replacing the service that the failed component had been providing. There are number of ways of doing this, called *cold standby*, *warm standby*, and *hot standby*.

Mode	In this standby mode:
cold	Repairing the service means noticing that the service has failed and bringing up a new module (i.e., starting an executable by loading it from media), initializing it, and bringing it into service.
warm	Repairing the service is the same as in cold standby mode, except the new the service is already loaded in memory, and may have some idea of the state of the service that just failed.
hot	The standby service is already running. It notices immediately when the primary service fails, and takes over. The primary and the standby service are in constant communication; the standby receives updates from the primary every time a significant event occurs. In hot standby mode, the standby is available almost immediately to take over — the ultimate reduction in MTTR.

Cold, warm, and hot standby are points on a spectrum:

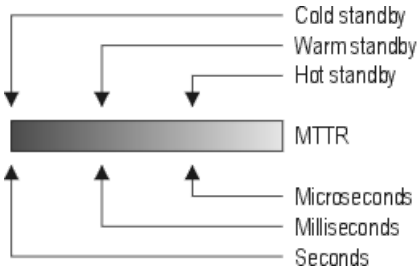

Figure 1. The MTTR spectrum.

The times given above are for discussion purposes only — in your particular system, you may be able to achieve hot standby only after a few hundred microseconds or milliseconds; or you may be able to achieve cold standby after only a few milliseconds.

These broad ranges are based on the following assumptions:

cold standby — seconds

I've selected "seconds" for cold standby because you may need to load a process from some kind of slow media, and the process may need to perform lengthy initializations to get to an operational state. In extreme cases, this scale could go to minutes if you need to power-up equipment.

warm standby — milliseconds

Milliseconds were selected for warm standby because the process is already resident in memory; we're assuming that it just needs to bring itself up to date with the current system status, and then it's operational.

hot standby — microseconds

Ideally, the hot standby scenario can result in an operational process within the time it takes the kernel to make a context switch and for the process to make a few administrative operations. We're assuming that the executable is running on the system, and has a complete picture of the state — it's immediately ready to take over.

Parent topic: Failure modes and recovery models