A Software System Health Invariant

Disclaimer: please reference this paper if you use or adapt this idea in your own work! I've prepared references for copy and pasting in the "How to Reference" section.

I've said it before (see attached "InvocationBasedRuntimeAnalysis.pdf") and I'll say it again: assessing the computational cost of a computer program requires analyzing the number of times various parts of the systems will execute and the characteristics of the data that will be passed to them. Simply zooming in to a single function call and looking at the rate of growth as the size of the input grows isn't instructive in most situations you'll face in practice.

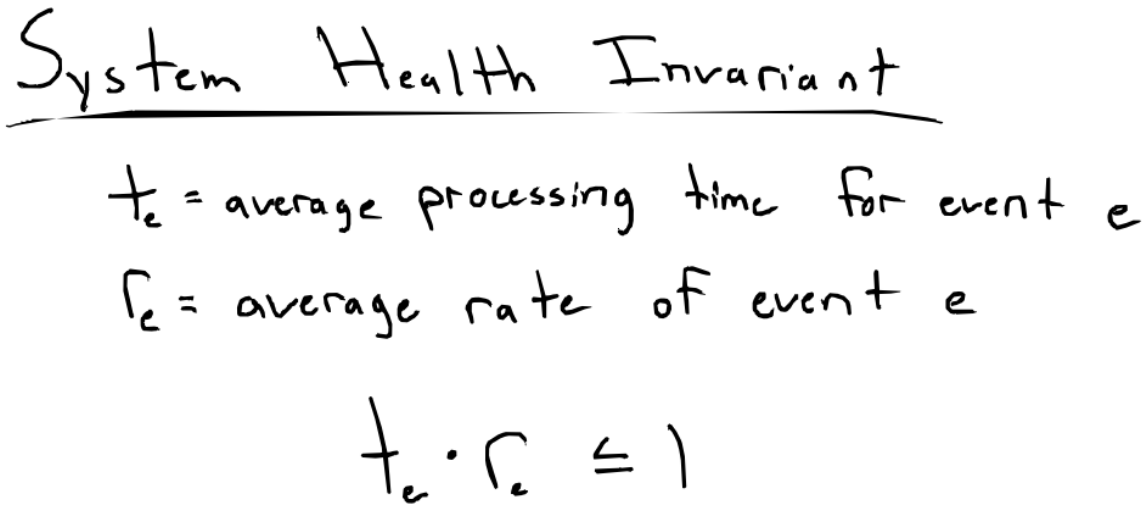
Setting aside the complexity that would be required to account for the characteristics of the data passed to a program, there is a pretty clear invariant you can derive from the insight above.

In natural language, it would be something like "if the time for a specific portion of the system to process an event is greater than the average time between two occurrences of the event the system will eventually fail".

I've used a lot of imprecise terms, so let me clarify. "Specific portion" would be any of the code in a system that must execute after an event occurs. There isn't a clear notion of this in programming or system design so I don't have a precise word to describe this. "to process" in this case just means any and all activity that happens on the computers the system is implemented on as a result of the event.

Finally, "event" can be any occurrence, machine or otherwise, that can eventually lead to a change in the state of the system and "fail" means cease functioning due to resource exhaustion (3).

Formally



Examples

Take, for example, something like a threat detection system. Threat detection systems process log data from computers in large networks and attempt to identify indicators of an attack in real time. It's not a stretch of the imagination to suppose a threat detection system may need to perform some sort of operation every time a user in the network logs in to a privileged account.

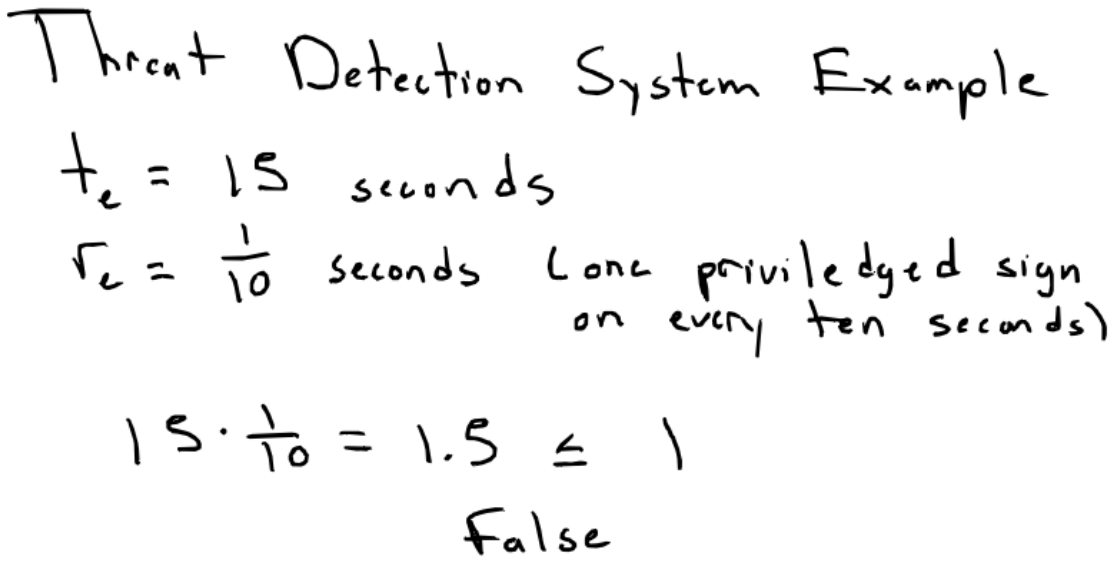
It may be inspecting the IP the user logged in from, checking against a list of bad IPs, profiling the user-agent, etc. There is an almost endless amount of things the system could do each time a user signs in to a privileged account to determine if there is an indicator of attack.

Suppose that you've created a threat detection system and, each time a user in the network logs in to a privileged account, it does a certain set of operations. Suppose you can observe via system telemetry that the entirety of the processing that must occur after a single privileged logon requires 15 seconds (1).

If it turns out that, as the company grows, or as more users gain access to privileged accounts, the average time between privileged account logons is 10 seconds, then it is likely that at some point the system requires more resources than are available to it and the hardware or software resources supporting the system will be exhausted and the system will fail.

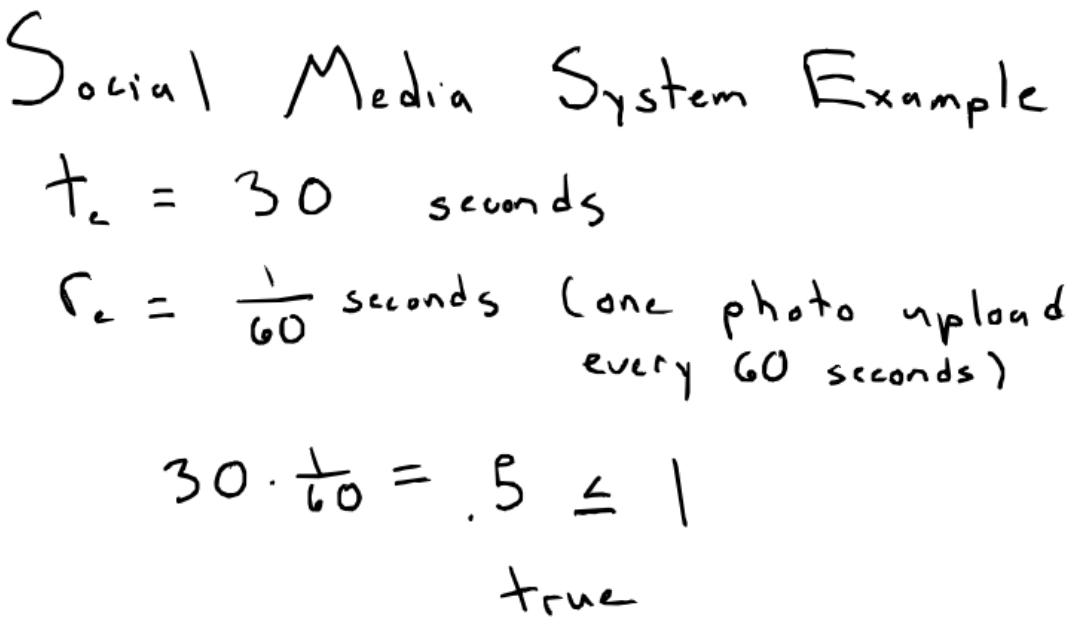
The only thing preventing a system failure would be luck or randomness. If you could map all of the events that result in some portion of the system executing, and you could measure the time the system requires to process each one (4), you could derive strong

guarantees of system health and remove a lot of guesswork in terms of establishing proper resource requirements for the hardware and software supporting the system.



Another example could be a social media system which allows users to upload photos to a public page (5). If the system performs some series of checks to validate the photos don't contain explicit or inappropriate images, and that series of checks requires, on average, 30 seconds to perform, if the user base uploads photos at an average rate quicker than one photo every thirty seconds the system would be likely to experience a failure state at some point in it's operation, especially if the user base grew and the average rate of photo upload became significantly quicker.

If the average rate of photo upload was every minute, the system would not violate the system health invariant I'm proposing here.



Limitations

One limitation here is that this invariant is based in an "event-based" framework, which may not account for variations in the characteristics of the data passed to the system. The framework may still be accurate enough, even when there is variation in the characteristics of data, given that is uses an average cost per event. It may also be that using an "event-based" framework is sufficient regardless as in practice variation in the characteristics of data passed to a system less likely to occur (2).

I suspect you may be able to reduce the characteristics of data to events themselves, although I haven't explored this possibility much.

How to Reference

I would appreciate it if you include a reference to this blog if you refer to or adapt this idea in your own work. I haven't included references below because as far as I can tell I didn't really use any in the deriving of this work. You can use the citation below:

Footnotes

1) It isn't necessarily impractical to imagine you could get a measurement of this kind in practice, many threat detection systems are query based or integrated with log-monitoring systems that have robust performance benchmarking.
2) This seems unlikely for generic systems and more likely for specialized systems.
3) In practice this may look like a system "freezing" (in the case of an operating system running out of RAM), becoming untenably unresponsive (in the case of a more complicated web-based application surpassing the constraints imposed by it's infrastructure) or simply failing to perform a certain function after reaching a timeout.
4) By "time to process" I mean the amount of time that is passing in which the instructions in the specific portion of the system are executing on a machine. This isn't necessarily the same as "time that has passed since the event occurred" since the system may have some sort of deferred or asynchronous processing components. You could also consider the invariant I'm proposing in the context of "time that has passed since the event occurred" although this would mean the invariant is more a measure of responsiveness and not necessarily system health.
5) I haven't worked on social media systems so please excuse me if this is an excessively artificial example.