Numbers Are for Computers, Strings Are for Humans

HOW AND WHERE SOFTWARE SHOULD TRANSLATE DATA INTO A HUMAN-READABLE FORM

Dear KV,

While debugging a set of networked systems that seemed to be executing operations out of the expected order, I discovered an interesting feature of the protocol used by the application. The system is fairly old, and I wasn't involved in its creation, but I was asked to figure out why about 10 percent of the transactions were flagged as being in the wrong order. All the application communication happens using TCP. And since TCP guarantees the ordering of messages, I was confused as to how transactions between two systems could arrive out of order. What I found—by using Wireshark—was that the TCP stream was, as expected, in order, but the application protocol used on top of TCP had some rather odd properties.

In particular, all of the information, including time, was communicated as strings. The bug turned out to be an incorrect conversion of the time from a string to a value that could easily be compared. Although the messages arrived in the correct order, the system, reading the time, thought they were out of order and complained loudly. When I finally tracked down the developer who wrote the code, he said that he had used strings to make the protocol easier to debug and to make it easier for people looking at the log file to know what was happening in the system. My feeling is that he got this concept the wrong way around,

and I'm wondering how you might feel about this, as you've written about time in the past.

Stung by Strings

Dear Stung,

How does this make me feel? Well, like nearly all questions around software and technology, it makes me angry, but then, what doesn't make me angry? When you play only one note, you might as well play it really well, even if your instrument is a hammer and anvil.

As you point out, the developer definitely has this the wrong way around for at least two reasons: The first has to do with how one communicates a value as important as time; and the other has to do with how and where software should translate data into a human-readable form.

Computer systems generally, and networked systems specifically, often depend on time and time stamps to maintain or reconstruct the ordering of events, and this ordering often must be maintained so that the system as a whole can function properly. Some systems can operate without needing to know the time of day; instead, they depend on a total ordering of events, as established by Leslie Lamport, who ACM saw fit to honor with a Turing Award for bothering to figure out that little problem (https://amturing.acm.org/award_winners/lamport_1205376.cfm).

There remains a class of systems that actually do care about the very human time of day, such as credit card processing. If your payment to the bank isn't timestamped with the appropriate time of day (e.g., midnight on the first day of the month), then you are going to be subject

to interest and penalty payments that will likely make you even angrier than KV. Time also plays an important role in many nonbanking security protocols. Get a comparison backwards, or let it roll over to 0 or to a negative value, and the security of the system is broken. The number of attacks on systems based on time fills a large number of papers and books on computer security.

The number of attacks on systems based on time fills a large number of papers and books on computer security.

Computers, it should be pointed out, like to work with numbers. Strings are for humans, and so the idea of storing something as integral as time in a string is ridiculous on the face of it. The only way to know if X came before Y with a string-formatted time is to convert the times into something easily comparable (i.e., integers) and then to have the computer do the comparison by math, which computers are very good at.

The number of times that a human will be looking at any of this data to make the same comparison is minuscule, which is the whole reason that pretty much all computer languages have a time data structure that can be both easily compared and that, hopefully, is resistant to misinterpretation. Not that we can't get time wrong as an integer; we can, but it's far less likely than getting it wrong when stored as a string and converted before the math happens. KV is a pretty permissive guy, but when it comes to time, he's pretty strict. Find your language's provided time type and use it, and check for errors on every comparison.

The second fallacy under which your software was written has become more common as computers have become more powerful, and that fallacy is that compute power should always be used to give the human the best

experience. When computers were slow and expensive, programmers were able to avoid dealing with the human question by pointing out that storing data in a compact numerical form resulted in better efficiency and use of an expensive resource. As computers became cheap and pervasive, many people pointed out that these efficiencies were no longer as strictly necessary as they used to be.

Those of us who continue to program near bare metal have never really let go of this cherished orthodoxy, but KV must, grudgingly, very grudgingly, admit that the other camp might have a point here. A few bytes here, a few instructions there, they do add up, but often saving them isn't worth the effort, unless that leads either to incorrect results (see previous section) or to increased complexity, which usually leads to incorrect results (again see previous section).

Unless what you are processing, storing, or transmitting are, quite literally, strings that come from and are meant to be shown to humans, you should avoid processing, storing, or transmitting that data as strings. Remember, numbers are for computers, strings are for humans. Let the computer do the work of presenting your

Related articles

- Time is an illusion
  Lunchtime doubly so.
  George Neville-Neil
  https://queue.acm.org/detail.cfm?id=2878574
- Time, but Faster
  A computing adventure about time through the looking glass
  Theo Schlossnagle, Circonus
  https://queue.acm.org/detail.cfm?id=3036398
- There is No Now
  Problems with simultaneity in distributed systems
  Justin Sheehy
  https://queue.acm.org/detail.cfm?id=2745385

data to the humans in a form they might find palatable. That's where those extra bytes and instructions should be spent, not doing the inverse.

KV

George V. Neville-Neil *works on networking and operating-system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who currently lives in New York City.*