TSC-I2: A Lightweight Implementation for Precision-Augmented Timekeeping

Xun Luo, University of Illinois at Chicago, xluo@cs.uic.edu

I. Introduction

The quality of timekeeping is critical for many network protocols and measurement tools. Software packages, such as UDT[1], thrulay[2] and owamp[3], make control decisions and network statistics by checking the timestamps on the sent/received packets, which are normally labeled according to the system clock. Incorrect and/or imprecise system clock will inevitably bring negative effects to them, more or less.

Two metrics can be defined for the quality of a clock: accuracy and precision. Accuracy indicates the bound of a clock's offset, comparing with a correct reference; Precision indicates the resolution, regardless of its reporting resolution. Although when synchronized with NTP [4], system clock could achieve satisfying accuracy, its precision is still a problem at microsecond level. For example, most Linux kernels update their system time counters at 10-millisecond interval, and get the time in-between by interpolating TSC register. As interpolation parameters are obtained at start-up and TSC register subject to frequency wander, the interpolated time's precision is questionable.

II. Key Features of TSC-I2

TSC-I2 (TSC-Internet2) is implemented to address the precision problem mentioned above. The basic idea is to make the TSC rate calibration a continuous process, thus the accuracy of interpolation parameters could be ensured, which in turn results in satisfying clock precision. TSC-I2 maintains a soft clock of its own, compares this clock to system clock periodically. During each comparison, it synchronizes itself with the system clock, and adjusts the interpolation rate based on the offset and rate errors regarding to system clock. Whenever the accuracy of the soft clock is ensured, TSC-I2 uses this clock to report time to the library user; otherwise, the system clock value is reported.

Key features of TSC-I2 include:

1) Sys-clock augmentation, rather than substitution

Only when TSC-clock is in synchronization with system clock its value is reported; otherwise TSC-I2 wraps system clock without significant overhead.

2) Quick convergence, with high rate accuracy

A state-machine-controlled PLL (Phase Lock Loop) traps the rate-induced phase difference between TSC-clock and system clock. Rate wander is captured within one loop delay, and corrected in 3 to 4 following loops.

3) Robustness to spikes and system noises

Two filters figure noise out of rate wander. A popcorn filter picks out offset outliers based on the running standard deviation test; a spike-detector in the state machine further filters spikes from the time adjustments.

4) Flexible running options

Both DAEMON mode and CLIENT mode are supported by TSC-I2. In DAEMON mode, a standalone daemon takes charge of timekeeping, serving one or more clients. In CLIENT mode, the library creates a thread running within the host process. Thus it minimizes the application's external dependency.

5) Light weight and resource efficient

TSC-I2 consumes less than 0.5% CPU time when fully functioning; and the whole package only uses 500k disk space.

III. Design Of the Library

Fig. 1 sketches the process of TSC-I2. The library first synchronizes itself with the system clock in offset. So at that instant, it's reporting the same time value with system clock.

Right after the synchronization, the library gets into rate converge phase. In this phase, the library adjusts its interpolation parameters (especially rate), apply it on TSC to calculate a time value, and compare this value with the reading of the system clock. If those two match, the interpolation parameters could be regarded as correct. Otherwise, the rate is treated as "unstable yet", and the phase is repeated. To avoid wrong interpolation parameters lead correct time value by luck, the rate stable test has to succeed at least three times before safe assertion could be made.

Once the rate is also synchronized, the issue is to continuously keep this, and adapt to any TSC rate wander. The PLL observes soft clock reading and system clock reading during each filter loop; and makes corresponding rate adjustment to correct future offset error.

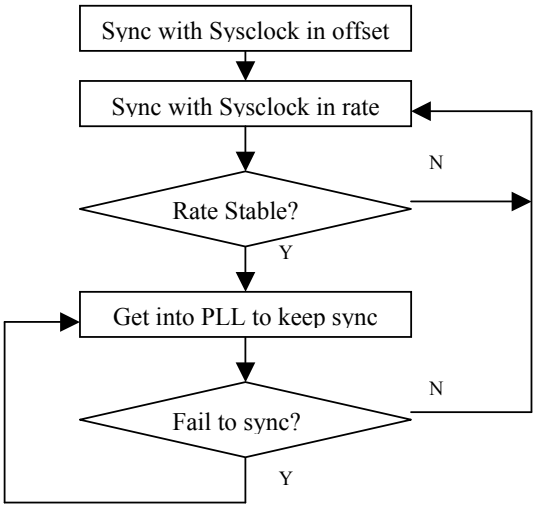

Fig. 1 Flow chart of TSC-I2 internal process

IV. Current Status

TSC-I2 has been published under Open Source License at http://tsc-xluo.sourceforge.net. Present release is 0.08, which includes a user-mode daemon tsci2d, a C library libtsci2 and a set of utility tools as tsci2demo and tsci2measure. TSC-I2 currently supports IA32, AMD64 and Power PC

architectures, as well as Linux, FreeBSD, Mac OSX and Microsoft Windows operating systems.

V. Acknowledgements

I would like to thank Google Inc., who initiated the "Summer of Code" event to promote open source spirit, and helped participating students and their mentoring organizations to match each other based on common interests. The job of organizing a 400-project event successfully is great.

I owe much to my mentoring organization, Internet2, and its staff Stanislav Shalunov and Jeff Boote. They helped me and our 10-people "Internet2 group" greatly in many aspects: managing the mail list; holding office hours; answering technical questions; reviewing codes; testing programs; giving critiques and suggestions; and providing software and hardware supports. Their technical excellence and generous effort contribution is key to this project. I sincerely thank them.

Thanks for all the fellow team members in "Internet2 group" for active discussion, idea sharing and helping to test programs. This is the first team I have been in which is purely internet-connected. The experience is such a fun and will surely benefit me in the future.

References

[1] http://udt.sourceforge.net/.
[2] http://www.internet2.edu/~shalunov/thrulay/.
[3] http://e2epi.internet2.edu/owamp/.
[4] http://www.ntp.org/