TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Continuous Clock Synchronization for Accurate Performance Measurements

Motivation

Clock synchronization is a key prerequisite for accurate performance analysis. This is particularly true for most HPC systems, due to the lack of a system-wide timer. The effects of different clock properties lead to false measurements and wrong conclusions. Often these errors remain undetected by the user. Different environmental effects cause continuous changes to clock properties over time. This behavior is ignored by the usual post-mortem approach for clock synchronization. In order to improve time measurement accuracy, we implemented a method for continuous clock synchronization. In this work we share our experiences and draw conclusions for tool development.

Challenges

Clocks usually emit time in a monotonic increasing way, using a certain resolution that needs to satisfy the required measurement accuracy.

Clocks in HPC machines may suffer from multiple effects:

- Considerably large timestamp differences due to different starting points
- Changes due to NTP or sys-admin corrections
- Differences in speed resulting in a measurable drift
- Acceleration and deceleration due to fluctuations of temperature or voltage

The described effects may alter clock behavior only in the nanoseconds range, but are still measurable and accumulating over time.

One-time as well as post-mortem synchronization approaches are insufficient as changes can vary over time.

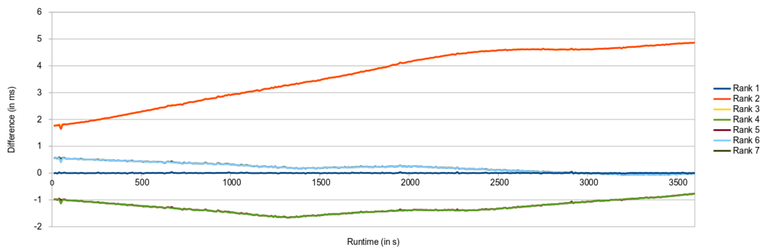

Fig. 1 Uncorrected clock differences to Rank 0 – measured with 8 processes on 4 nodes.

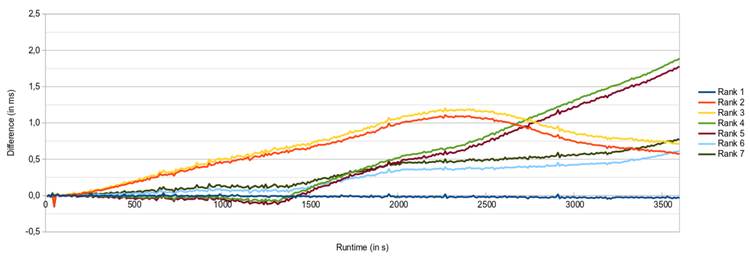

Fig. 2 Clock differences to Rank 0 after a single synchronization step in the beginning.

Continuous Sync of Two Clocks

The basic idea for synchronizing two clocks is to determine the required correction factors (intercept and slope) for adjusting the local time of a client process to a global synchronous time. These factors are computed using *send* (t_0, t_2) and *receive* (t_1, t_3) timestamps of injected messages between the two processes.

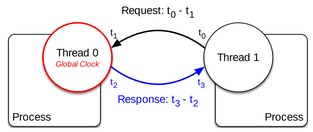

Fig. 3 Synchronization setup for two clocks. *Thread 1* synchronizes its time to the clock of *Thread 0*. Transfer times (Offset in Fig. 4) of *request* and *response* messages are the basis for the computation of the required time correction factors (intercept and slope).

The ***Intercept*** value describes the absolute time offset between two clocks. It is derived from the difference (offset) between the send/receive timestamps of the injected messages. This value inherently includes the messages transfer time. However, this fact is neglected as it is impossible to distinct both. The message with the lowest offset describes the most accurate intercept value.

The ***Slope*** value describes the drift between two clocks. The larger the slope, the faster the times of two clocks run apart. The slope may change continuously over time. The slope value is necessary to extrapolate the current offset from the position of the last lowest latency message.

Due to the large amount of offset measurements a linear regression is not feasible for the computation of the slope. Thus, a *convex hull* approach is applied to select only low latency messages (providing the highest accuracy) for the computation. Two convex hulls are computed, one for each message direction (request and response) independently. The tangent between both convex hulls defines the slope value.

Intercept and slope are computed based on a sliding window technique. The width of the window controls the sensitivity/stability of the synchronization approach to clock changes.

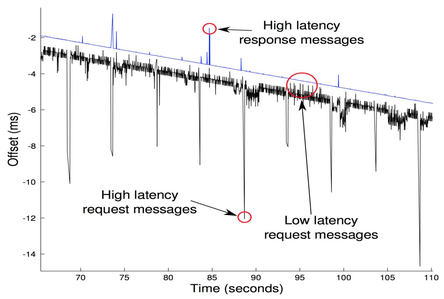

Fig. 4 Request (black/bottom) and response (top/blue) message offsets. The two lines bound the true offset. Request messages typically have higher offsets (latency) than responses.[1]

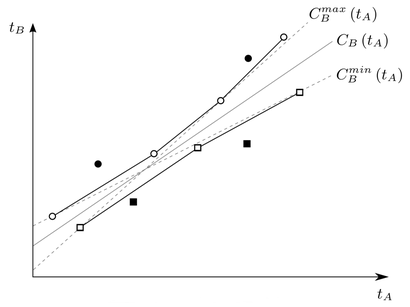

Fig. 5 Selecting low latency messages is achieved by computing the convex hull for each message direction ($C_B^{max/min}$). The tangent C_B between C_B^{max} and C_B^{min} defines the slope. [2]

Multi-stage Sync of Multiple Clocks

Extending the pairwise approach allows to synchronize multiple clocks. One option is to define one clock as global clock and let all other clocks synchronize their time with the global clock in a master-slave fashion. However, this approach is not scalable and may lead to overloading of the global clock process. A more scalable option is to follow a multi-stage approach. In that case clock pairs are arranged in a tree-like hierarchical fashion. The clock at the root level defines the global time. Following the hierarchy, clocks synchronize their time with clocks in the above hierarchy layer. Fig. 6 shows the synchronization scheme for four clocks using a two level hierarchy.

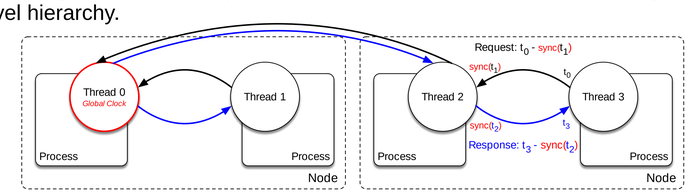

Fig. 6 Two-stage synchronization of four clocks. *Thread 0* defines the global time. In the first stage *Thread 2* synchronizes its time with *Thread 0*. In the second stage *Thread 1* and *Thread 3* synchronize times with their respective partner.

Conclusions

Continuous clock synchronization is mandatory for accurate measurements on most HPC systems. Our approach greatly improves measurement accuracy especially for long running applications. Experiments show that our prototype implementation provides accurate timings in the range of one millisecond even for several hours of application runtime. We found that a sliding window length of 5 min provides the best overall stability/sensitivity. We extend upon previous work by employing a hierarchical synchronization scheme that can handle large numbers of clocks. Our work has shown that time synchronization methods based on intercept and slope computation are applicable in the context of performance measurements of parallel applications.

[1] A. Harrison and P. Newman. TICSync: Knowing When Things Happened. In ICRA. 2011.
[2] Benjamin Poirier, Robert Roy, and Michel Dagenais. Accurate Offline Synchronization of Distributed Traces Using Kernel-Level Events. SIGOPS Oper. Syst. Rev. 2010.

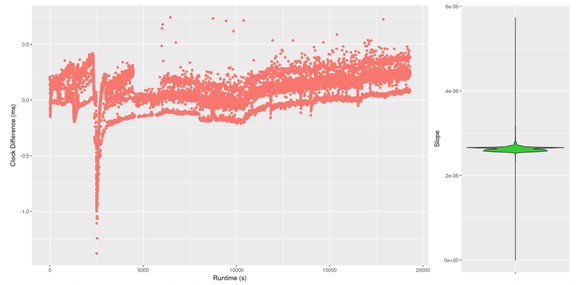

Fig. 7 Synchronization of two clocks using our prototype. Measurement of the difference between the global time and the synchronized time. Both clocks differ in the range of about 1 *ms*, already including the error for the message transfer time.

Johannes Ziegenbalg (johannes.ziegenbalg@tu-dresden.de), Matthias Weber (matthias.weber@tu-dresden.de)
Technische Universität Dresden, ZIH
Germany

Center for Information Services & High Performance Computing

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur