# Evaluation of NTP/PTP fine-grain synchronization performance in HPC clusters

**Conference Paper**

**Author(s):**
Libri, Antonio; Bartolini, Andrea; Cesarini, Daniele; Benini, Luca

**Publication date:**
2018-11-04

**Permanent link:**
https://doi.org/10.3929/ethz-b-000306928

**Originally published in:**
ACM International Conference Proceeding Series, https://doi.org/10.1145/3295816.3295819

# Evaluation of NTP / PTP Fine-Grain Synchronization Performance in HPC Clusters

Antonio Libri
IIS, ETH Zurich
Zurich, Switzerland
a.libri@iis.ee.ethz.ch

Andrea Bartolini
DEI, University of Bologna
Bologna, Italy
a.bartolini@unibo.it

Daniele Cesarini
DEI, University of Bologna
Bologna, Italy
daniele.cesarini@unibo.it

Luca Benini
IIS, ETH Zurich
Zurich, Switzerland
lbenini@iis.ee.ethz.ch

## ABSTRACT

Fine-grain time synchronization is important to address several challenges in today and future High Performance Computing (HPC) centers. Among the many, (i) co-scheduling techniques in parallel applications with sensitive bulk synchronous workloads, (ii) performance analysis tools and (iii) autotuning strategies that want to exploit State-of-the-Art (SoA) high resolution monitoring systems, are three examples where synchronization of few microseconds is required. Previous works report custom solutions to reach this performance without incurring in extra cost of dedicated hardware. On the other hand, the benefits to use robust standards which are widely supported by the community, such as Network Time Protocol (NTP) and Precision Time Protocol (PTP), are evident. With today's software and hardware improvements of these two protocols and off-the-shelf integration in SoA HPC servers no expensive extra hardware is required anymore, but an evaluation of their performance in supercomputing clusters is needed. Our results show NTP can reach on computing nodes an accuracy of 2.6 µs and a precision below 2.7 µs, with negligible overhead. These values can be bounded below microseconds, with PTP and low-cost switches (no needs of GPS antenna). Both protocols are also suitable for data time-stamping in SoA HPC monitoring infrastructures. We validate their performance with two real use-cases, and quantify scalability and CPU overhead. Finally, we report software settings and low-cost network configuration to reach these high precision synchronization results.

## KEYWORDS

NTP, PTP, HPC Clusters, MPI, Fine Grain Synchronization, Power and Performance Monitoring

## 1 INTRODUCTION

Time synchronization is a key factor for any distributed system [13]. This is especially true for High Performance Computing (HPC) clusters, considering the trend toward increasing the node count to exploit application concurrency and parallelism. In this context, it becomes crucial to ensure a tight level of time agreement [5, 13, 14]. As reported in [13], this is needed for several reasons, with the ultimate goal to improve the application execution time and energy efficiency of the system (*i.e.*, *Time-to-Solution* - TtS - and *Energy-to-Solution* - EtS). As example, a fine-grain synchronization (order of few µs) is fundamental in (i) wall-clock runtime of parallel applications that exploit the Bulk Synchronous Parallelism (BSP), which is a parallel programming model where the progress of concurrent processes is driven by synchronization points (synchronization barriers). As shown in [13], by mean of a coordinated scheduling (co-scheduling) strategy it is possible to achieve a speedup of 285 % for sensitive bulk synchronous workloads (*i.e.*, with frequent synchronizing collectives based on Message Parallel Interface - MPI -, such as MPI_Barrier or MPI_Alltoall).

Other examples are (ii) code development / debugging and (iii) allow autotuning techniques to exploit new generation of high resolution monitoring systems [12, 13, 16]. Works in [4–6] show that synchronization performance in the order of few microseconds is a must to carry out an efficient analysis of applications' performance, as otherwise would not be possible to achieve a correct ordering of the events in concurrent processes running on multiple nodes. Moreover, SoA high resolution monitoring systems for HPC (e.g., DiG, HAEC [12, 16]) require a synchronization at least comparable to their sampling rate, to be able to correlate the measurements with application phases, and therefore to get a detailed profiling picture over time needed for a better usage of the resources.

Time agreement in distributed systems is normally supported by network synchronization protocols. In these protocols all the nodes are kept synchronized against a time reference assumed as true time. In this context, accuracy refers to the amount of shift between the time reference and the mean of the measurements of time ($\mu$), and precision to the standard deviation ($\sigma$). Over the past years two standards have emerged and are widely adopted: Network Time Protocol (NTP) [7] and Precision Time Protocol (PTP) [1] (aka IEEE 1588). The former targets Wide Area Networks (WANs), where it typically achieves accuracy of milliseconds, but - as documented in the standard [7] - can reach better performance within Local Area Networks (LANs). PTP targets LANs and can synchronize devices with sub-microsecond performance, thanks to dedicated hardware support for synchronization.

In particular, within the context of HPC systems and data-centers, NTP is today the most used protocol [13], but - as reported in a recent survey on leadership class supercomputing centers [13] - is usually set with default configurations, obtaining a synchronization accuracy that does not meet acceptable uncertainty requirements for many potential applications (jitter of tens / hundreds of milliseconds). To achieve microseconds synchronization without incurring in extra cost for dedicated hardware support, such as with PTP, several works in literature proposed alternative software solutions [4–6, 14]. However, the advantages to use robust standards that are widely supported by the community - like NTP and PTP - are evident. In addition, (i) today's software / hardware implementation improvements of these two protocols (*e.g.*, enhancements on software deamons, high stability of commodity oscillators), along with (ii) off-the-shelf integration of PTP hardware support in State-of-the-Art (SoA) HPC servers and IoT platforms commonly used for embedded monitoring solutions, give a motivation for our work.

In this paper we focus on the evaluation of NTP and PTP synchronization performance in a SoA HPC Cluster - *i.e.*, D.A.V.I.D.E. [2] (18th in Green500 November 2017) - and its out-of-band monitoring infrastructure, namely DiG and Examon [3, 16]. A key concept in our study is that synchronization is performed within the LAN of the HPC infrastructure. This allows to work in a corner-case for NTP, where it can achieve its best performance. Moreover, it is not necessary to achieve high accuracy with respect to the absolute time reference (e.g., by mean of an atomic clock or a Global Positioning System - GPS - antenna), but instead that all devices are highly synchronized between them.

Major contributions of this paper are:

(1) the proof that with a proper configuration both NTP and PTP can achieve the target synchronization of sub-milliseconds needed for many potential applications in today's supercomputers. This is done through extensive measurements of synchronization's performance in a SoA HPC cluster and its out-of-band monitoring infrastructure. Moreover, we carry out a validation of the results in two real use-case scenario. We will show that NTP can achieve in HPC nodes an accuracy below 2.6 µs and a precision below 2.7 µs. Regarding PTP, these values are bounded below microseconds. It is noteworthy that is possible to achieve these performance with low-cost COTS hardware, which generally is already built-in in new supercomputers (no extra cost is required). Finally, we observed that the synchronization performance achievable in the IoT devices used in DiG (*i.e.*, best-in-class HPC high resolution monitoring) is below their sampling period (20 µs), thus is suitable to correlate the monitored metrics with applications running in the computing nodes;
(2) a detailed investigation of scalability and CPU overhead of the protocols' SW daemons distributed in the cluster devices;
(3) a comprehensive description of the network topology and SW configuration used to achieve our performance, with the goal to be helpful for other leadership class supercomputers.

*Organization of the paper:* we introduce in Section 2 an overview of D.A.V.I.D.E. and its out-of-band monitoring system, and the importance of a fine grain synchronization in HPC clusters (together with some background information of the two evaluated protocols). We outline the related work in Section 3. Section 4 reports our performance measurement results and a *how-to* guide that outlines our network topology and configuration. Finally, we conclude the discussion in Section 5.

## 2 HPC CLUSTERS AND FINE GRAIN SYNCH

On the race toward exascale computing, HPC systems are facing crucial challenges which limit their efficiency. Among the many, power and energy consumption, bounded by the end of Dennard scaling, start to show their impact on limiting supercomputers peak performance and cost effectiveness [2, 12]. This is evident by looking at the latest Top500 list (June 2018): the most powerful supercomputer - *i.e.*, Summit - has an improvement in energy efficiency of 2.3 × over the previous one - *i.e.*, Sunway TaihuLight -, but only an improvement in performance of 30 %. We are clearly in an era of power limited HPC evolution, driven by new hardware technologies along with a better usage of the resources.

At the basis of a better usage of the resources, techniques for exploiting more efficient concurrency on applications that run on many cores per chips (*i.e.*, CPUs / Accelerators) and multiple nodes per clusters are of crucial importance [13, 14]. Already several works in literature show that an efficient concurrency, which ultimately leads to improvements in TtS and EtS, can be achieved by (i) co-scheduling techniques [13], (ii) optimized coding, which requires application's performance analysis tools for debugging [6, 13], and (iii) runtime autotuning techniques, based on power and performance monitoring, to promptly react to workloads changing and events (e.g., by mean of Power Capping and Dynamic Voltage and Frequency Scaling - DVFS; Approximate Computing; advanced cooling control policies) [9, 20]. All these strategies require a proper level of synchronization within the HPC cluster. In particular, co-scheduling and performance analysis tools demand a synchronization accuracy in the order of few microseconds (4–7 µs [4–6]), while runtime autotuning, based on power and performance measurements, needs a synchronization accuracy lower than the sampling period of the monitoring, to be able to correlate application phases and events with physical and architectural metrics.

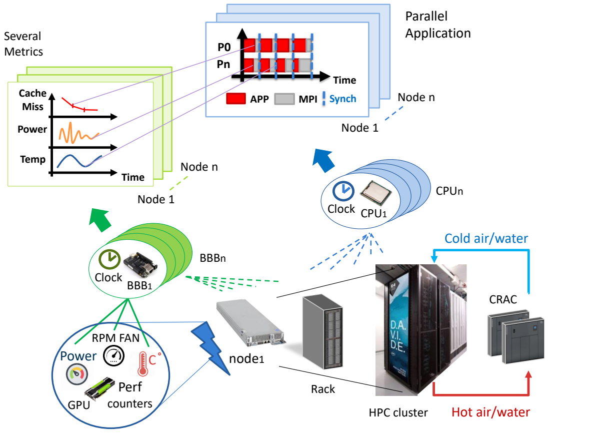


**Figure 1: Overview of D.A.V.I.D.E. and DiG, as example of SoA green HPC cluster.**

Figure 1 shows an example of a SoA energy efficient HPC cluster, namely D.A.V.I.D.E. [2, 3], and its monitoring infrastructure, DiG [16]. The cluster consists of 45 IBM OpenPOWER computing nodes, plus 2 management nodes, grouped in 3 racks. Each node includes 4 NVIDIA Tesla P100 and a dedicated embedded device (*i.e.*, Beaglebone Black - BBB - which is part of DiG) for an out-of-band monitoring up to 20 µs of sampling resolution and hardware support for PTP. Moreover, the system is liquid cooled and all the nodes exploit an EDR InfiniBand connection (100 Gbit/s). As depicted in Figure 1, to meet the accuracy synchronization constraints required by the aforementioned optimization strategies, it is important that the clocks within HPC nodes and embedded devices are tightly synchronized. These clocks are then used for co-scheduling strategies in applications running on multiple nodes, and to time stamp and correlate both (a) application phases / events and (b) measurements acquired by the monitoring devices.

Focusing on NTP and PTP, both use a client-server model, where each client regularly polls a cluster of servers to synchronize their clock [1, 7]. According to the standard nomenclature, the terms client / server are used for NTP, while slave / master for PTP. However, there is no conceptual difference between them (master corresponds to server and slave to client), and we will often use in this paper the terms master / slave for both protocols. The main difference between these protocols is the timestamp method to synchronize the clocks: while NTP can use only the so called *software timestamp*, which is taken at user-space and thus with more jitter (noise) that affects the synchronization, PTP can benefit of the *hardware timestamp* which is taken by dedicated hardware devices (*i.e.*, PTP Hardware Clock - PHC) with higher levels of accuracy and precision. In our target scenario, HPC nodes and embedded monitoring devices run the slaves clocks, while dedicated servers run the masters that provide the time reference for the whole network. In particular, the ultimate goal is to synchronize the *system clocks* of all the involved devices (each device has more than one clock, and the system clock is basically the main clock that provides the time for all other applications running in the device - e.g., to carry out co-scheduling techniques).

In Linux systems, NTP is usually implemented by the *ntpd* daemon, while PTP by two user-space applications, namely *ptp4l* and *phc2sys*. In particular, ptp4l is the actual implementation of PTP (*i.e.*, it synchronizes all PHCs within the network), while phc2sys is used to synchronize the PHC of each device with its system clock.

## 3 RELATED WORK

Widely used approaches to address time agreement in distributed systems consist of network synchronization protocols, such as NTP and PTP [1, 7]. The former is commonly used over WANs, where it can reach an accuracy of milliseconds, while the latter within LANs, for applications where higher synchronization performance is needed. However, PTP requires dedicated hardware which only recently started to be integrated off-the-shelf in common servers and embedded computers.

To achieve in the past years sub-millisecond synchronization without incurring in extra cost for dedicated hardware, prior works proposed alternative solutions. One technique which targets events synchronization in HPC parallel applications consist of post-mortem ordering [4–6]. This approach is useful for event tracing where application phases and the monitored metrics can be retroactively synchronized. However, it is not suitable for real-time synchronization, which is crucial to improve TtS and EtS of HPC applications [13].

To address this problem, Jones et al. [14] worked on a NTP-based synchronization scheme for MPI applications that provides an accuracy of 2.29 µs (min −32.0 µs, max 32.1 µs, span 64.1 µs). This solution is suitable for runtime synchronization of MPI applications, but not to timestamp and correlate measurements coming from monitoring systems. Another solution was proposed in [18], which consists of a software-based scheme called TSCclock that delivers up to ten microseconds of accuracy within a LAN. Even if this solution is suitable for several applications within distributed systems, this accuracy is still not enough for HPC synchronization constraints (*i.e.*, few microseconds). A more recent solution that provides higher synchronization performance is White Rabbit (WR) [19], which extends PTP with dedicated hardware, firmware and software, and can synchronize nodes with sub-nanosecond accuracy. However, as it is over requirements for the target constraints, considering also the extra hardware needed, we focus our analysis only on NTP and PTP.

To the best of our knowledge, only work in [10] carried out a similar evaluation, but exclusively for PTP and targeting different requirements. In particular, they focused on distributed systems where high performance synchronization against the absolute time reference was required. This is not needed to address HPC challenges (such as improving energy efficiency and execution time of applications) and at the same time can be an obstacle for supercomputing centers to carefully install a GPS antenna or an atomic clock. It should be noted that under these assumptions we performed in [15] an early evaluation of NTP / PTP performance, but only on IoT platforms (*i.e.*, BBB) and assuming point-to-point links between them (*i.e.*, no use of switches), thus measurements in a whole HPC cluster - *i.e.*, computing nodes, BBB and switches - are still missing.

## 4 EXPERIMENTAL RESULTS

In this section we evaluate the performance of NTP and PTP on a SoA HPC cluster, namely D.A.V.I.D.E. [2], in terms of (i) accuracy, (ii) precision, (iii) worst-case bound and (iv) scalability. Measurements are performed both on the computing nodes and on the out-of-band monitoring system of the cluster (DiG [3, 16], depicted in Figure 1). To interconnect all devices we used the *CISCO IE 3000* [8], a low-cost COTS switch, suitable for industrial installations and PTP HW-enabled.

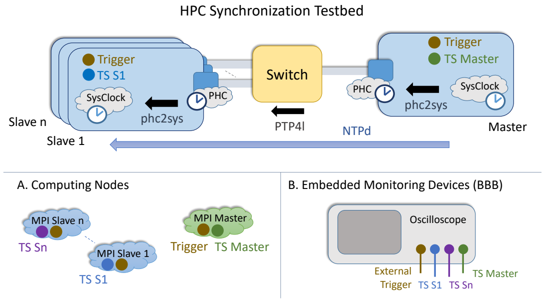


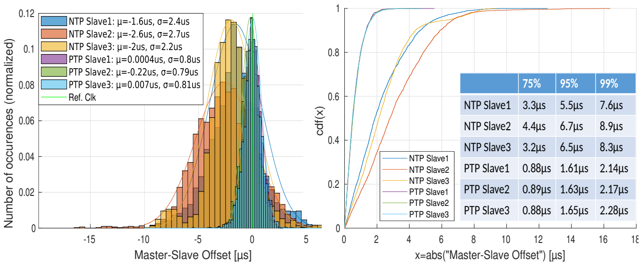
**Figure 2: Testbed used for NTP / PTP performance evaluation on computing nodes of D.A.V.I.D.E. and its out-of-band monitoring system (DiG).**

Figure 2 (top) illustrates testbed and synchronization flow of the SW daemons. Regarding NTP, we used ntpd to synchronize the system clock of the slaves directly to the system clock of the master node (ntp server), while for PTP we used ptp4l to adjust the PHC of the slaves to the master PHC and exploited phc2sys to constantly update the respective system clocks (on both master and slaves). Basing on the results obtained in our previous work [15] (focus of the paper was to find the optimal polling rate of NTP / PTP daemons to achieve best synchronization performance in a point-to-point link), we tuned their polling rate to the optimal operating point, which corresponds to 0.125 Hz (8 s) for NTP, and 1 Hz and 12 Hz for ptp4l and phc2sys, respectively. Moreover, we set the PTP switch in transparent mode to obtain higher levels of accuracy [8].

In the following, we will first evaluate (a) the synchronization between computing nodes, showing also an example with a parallel application, and then (b) the synchronization among monitoring devices, where we also verify the correlation performance between the out-of-band measurements and an application running on the computing nodes. Finally, for both protocols we will (c) quantify their scalability and CPU load on the devices, and (d) report a short how-to guide to setup the SW daemons and build the synchronization network to reach the performance reported in this paper.

## 4.1 Synch among HPC computing nodes

To measure the synchronization between computing nodes, we developed a synthetic benchmark based on *OpenMPI*, with one MPI process per node, as depicted in Figure 2.a. Goal is to generate timestamps on a set of queried nodes after a triggering event, and observe the clock's skew of the clients from the time reference. The benchmark first gets the hostname of the machine where is executed, then waits for an *MPI_Barrier* to align the timestamps generation events between nodes, and finally produces the timestamp using the *clock_gettime* function. As in common supercomputer installations the MPI inter-node latency for small messages is in the order of few microseconds (*i.e.*, accuracy within 4–7 µs [4–6]), it is crucial that clock synchronization of computing nodes is in this range or below.



| | 75% | 95% | 99% |
|---|---|---|---|
| NTP Slave1 | 3.3µs | 5.5µs | 7.6µs |
| NTP Slave2 | 4.4µs | 6.7µs | 8.9µs |
| NTP Slave3 | 3.2µs | 6.5µs | 8.3µs |
| PTP Slave1 | 0.88µs | 1.61µs | 2.14µs |
| PTP Slave2 | 0.89µs | 1.63µs | 2.17µs |
| PTP Slave3 | 0.88µs | 1.65µs | 2.28µs |

**Figure 3: NTP / PTP performance on computing nodes.**

We run the benchmark to generate 10 thousand timestamps over several hours on the nodes of D.A.V.I.D.E. synchronized first with NTP and then with PTP. Figure 3 (top) shows the histogram of the obtained master-slave offset, where the green line on the zero represents the time reference (*i.e.*, NTP / PTP server), and the y-axis reports the relative number of occurrences (number of observations in bin / total number of observations). Results show a clear Gaussian trend, for both NTP and PTP. In particular, accuracy ($\mu$) and precision ($\sigma$) are below 2.6 µs and 2.7 µs in NTP, respectively, and below 0.2 µs and  0.8 µs in PTP. To observe how the percentage of samples skews from the reference time, we report in Figure 3 (bottom) the Cumulative Distribution Function (CDF). NTP bounds 75 % of the samples within 4.4 µs, while PTP within 0.8 µs. These values increase up to a maximum of 8.9 µs and 2.2 µs for 99 % of the samples. It should be noted that D.A.V.I.D.E. exploits an EDR InfiniBand connection (100 Gbit/s) between nodes with a minimal latency of around 0.5 µs [17]. As this latency affects the MPI triggering event to generate timestamps on the nodes, both synchronization protocols would perform even better than what we measured (we are providing worst-case bounds). In light of these results, the time agreement provided by both protocols (*i.e.*, 2.6 µs of NTP accuracy, and 0.2 µs of PTP accuracy) is suitable to appreciate and correlate parallel application phases running on different nodes, as well as to correlate them with in-band system performance metrics timestamped up to few microseconds granularity.

In particular, the accuracy provided by NTP on computing nodes is an important off-the-shelf achievement for HPC supercomputing centers, researchers and developers, as it can push the boundaries of SoA techniques for profiling, debugging, scheduling and maintenance, and thus for improvements on HPC systems and applications performance [4–6, 13, 14]. To show one of the benefits of such a fine grain synchronization, we report a common scenario that can be found by HPC application's developers and users, when they have to balance the workload among MPI processes. The example consists of a scientific parallel application, namely *Quantum Espresso* (QE) [11], which involves several *MPI_Alltoall*. Let's assume the user wants to evaluate the unbalance in the application communication. QE involves several MPI_Alltoall on which all MPI processes wait on a barrier for the other processes to end their computation. We want to compute the time each MPI process takes to exit the barrier after the last process reaches it. We run QE on two computing nodes - one executing an MPI root and one an MPI slave - both synchronized to a time reference on a third node (*i.e.*, NTP server).

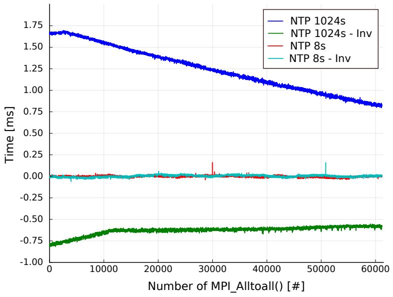


**Figure 4: Master-slave MPI_Alltoall offset with NTP polling rate set at 1024 s and at 8 s.**

Figure 4 shows the offset of the MPI processes at the exit of the barrier, for each MPI_Alltoall call in the application. In particular, the blue curve reports the offset when the nodes are synchronized with the default NTP polling rate (i.e, 1024 s, which is the one commonly used): the offset is in the order of few milliseconds and is linearly improving over time up to 0.75 ms. This result can be misleading, as it could mean there is an increasing communication unbalance (*i.e.*, MPI slave always leaves the barrier first) and latency for which it would be worst to tweak the code to alleviate its cost.

However, this is not the case, as if we keep the same workload but invert the nodes that are executing QE (green trace in Figure 4), we obtain exactly the opposite behaviour, where MPI root always leaves the barrier first. This is a clear problem of clock's synchronization and is much more challenging to understand when facing with thousands / millions of processes in large-scale supercomputing centers. In particular, the clock of a node is ahead of time in respect to the other node and the linear improvement of the offset is due to the NTP adjustments over time (jitter in the order of milliseconds because we are using the default polling rate). As shown by the two curves *NTP 8 s* and *NTP 8 s - Inv*, when we repeat the previous tests synchronizing the nodes with best settings (*i.e.*, NTP polling rate of 8 s), MPI processes exit the barrier with an offset roughly bounded within 80 µs. Due to the microseconds granularity of MPI processes, this fine grain synchronization - that can be obtained off-the-shelf - can easily and drastically improve application's debugging and tuning, crucial to increase efficiency of HPC systems.

## 4.2 Synch of the out-of-band monitoring

An accurate synchronization of the D.A.V.I.D.E. out-of-band monitoring infrastructure (DiG) is crucial to get a high quality profiling over time of the whole HPC system and running applications. As the maximum sampling period of DiG is 20 µs, it is important that its synchronization accuracy is in this range or below to be able to correlate out-of-band measurements between them (*i.e.*, the measurements acquired for each server would be perfectly aligned) and with application phases running on computing nodes. To evaluate the DiG synchronization performance we used the same testbed of before - same switch and software daemons on the embedded monitoring devices -, but with an external source of interrupts in order to get the same triggering event on all BBB (*i.e.*, computing platforms on which DiG is built upon). In particular, we used a square wave signal connected to the General Purpose Input/Output (GPIO) pins of the BBB. The GPIOs are handled by a custom device driver: as soon as the square wave goes high, an Interrupt Service Routine (ISR) in each BBB catches the event and generates a timestamp. To take into account the jitter related to the ISR processing time, we used another GPIO connected to an oscilloscope (*Keysight DS0X3054T*, depicted in Figure 2.b). After the timestamp is generated, we raise up the second GPIO and compare the ISR's jitter of the BBB master with the one of each BBB slave, obtaining the $\Delta_{ISR}$, as outlined in Figure 5. Thus, to evaluate the master-slave offset ($MS_{off}$), we used the following formula:

$$MS_{off} = TS_s - (TS_m + \Delta_{ISR}) \tag{1}$$

where $TS_s$ and $TS_m$ correspond to the system clock timestamps of the BBB slave and master, respectively. Moreover, the offset $\Delta_{ISR}$ is always referred to the master and can be either positive or negative depending on which clock is ahead of time.

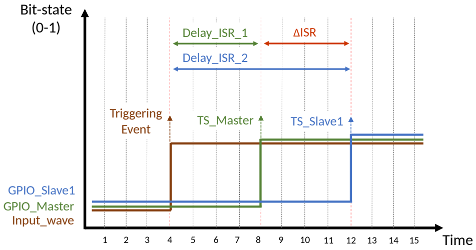
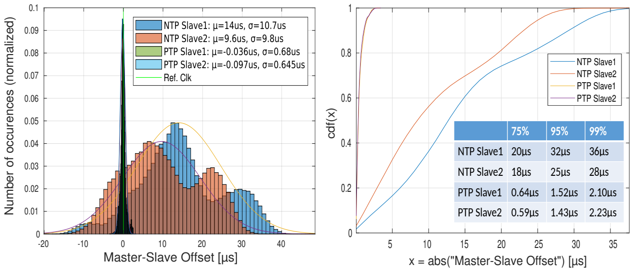


**Figure 5: Oscilloscope's window showing the delays between the triggering event and ISR processing time of each BBB (*i.e.*, Delay_ISR_1 and Delay_ISR_2).**

For the scope of this test, we synchronized 2 BBB slaves to a BBB Master, used as time reference, and generated 30 thousand timestamps. Figure 6 (top) shows the master-slave offset, where as before the green line on the zero represents the time reference and the y-axis the relative number of occurrences. Results of NTP show an almost Gaussian trend with an accuracy within 14 µs and a precision below 10.7 µs. The Gaussian curve becomes much more tighter for PTP, where we see an accuracy below 0.1 µs and a precision within 0.68 µs. To observe how the percentage of slaves' timestamps skew from the master, we report the CDF in Figure 6 (bottom): 75 % of the occurrences stay within 20 µs for NTP and 0.64 µs for PTP. These values increase up to a maximum of 36 µs (NTP) and 2.23 µs (PTP) for 99 % of the samples. Comparing these results with those obtained in the computing nodes, we can observe that PTP slightly improves its performance, while NTP is slightly worse (probably due to a cheaper built-in oscillator). However, we can state that both protocols are suitable to keep aligned the out-of-band power and performance measurements of D.A.V.I.D.E..



| | 75% | 95% | 99% |
|---|---|---|---|
| NTP Slave1 | 20µs | 32µs | 36µs |
| NTP Slave2 | 18µs | 25µs | 28µs |
| PTP Slave1 | 0.64µs | 1.52µs | 2.10µs |
| PTP Slave2 | 0.59µs | 1.43µs | 2.23µs |

**Figure 6: NTP / PTP performance on BBB.**

To verify the correlation performance of the DiG's measurements with an application running on computing nodes, we developed a custom benchmark based on *OpenMP*. The benchmark simply get a timestamp and stresses all CPU cores, while the nodes and the embedded monitoring devices are synchronized with NTP (same testbed of before, see Figure 2). The result is visible in Figure 7, where the blue curve corresponds to the power consumption monitored by DiG (sampling period of 20 µs, represented by the stars) and the red line to the timestamp generated by the application, which highlights the instant of rising edge: if we look at it, we can see it is well aligned with the power consumption trace. With this test we can finally validate both synchronization protocols to achieve a fine-grain synchronization in SoA HPC clusters.

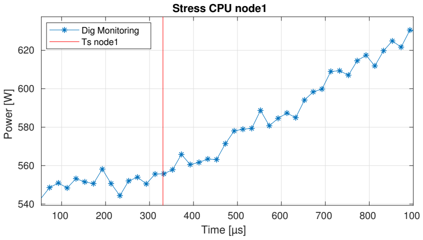


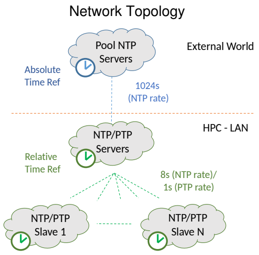
**Figure 7: OpenMP synthetic benchmark to validate the correlation performance between D.A.V.I.D.E. nodes and DiG.**

## 4.3 Scalability and CPU load

*Scalability.* Looking at the message synchronization pattern of both protocols [1, 7], we can quantify how the master (bottleneck) scales with the number of connected devices. Regarding NTP, considering N clients, the master has to deal with 2N packets (a pair query-reply) per clock's time update: with a polling period of 8 s this corresponds corresponds to 23 B/s per client. About PTP, using the Multicast communication model and only transparent clocks [1], the master has to handle 2 + 2N packets per clock's time update. In other words, setting a polling period of 1 s we have a fixed component of 180 B/s and a scalable component of 186 B/s per slave [1, 15]. Accordingly to these data rates, both protocols are not critical in terms of scalability. As example, considering a *Fast Ethernet* with a bit rate of 100 Mbit/s (e.g., 10/100 RJ45 on BBB), an NTP server would use only 0.000 184 % of the network bandwidth per slave, and a PTP master only 0.001 488 %. In theory, using only 10 % of the bandwidth an NTP server could handle up to 54k nodes, while each PTP master up to 6.7k nodes. Moreover, using a *Gigabit Ethernet* these values improve of an order of magnitude (e.g., 540k nodes for each NTP server and 67k nodes for each PTP master).

*CPU load.* In order to do not impact on the HPC computing resources, it is important that the proposed configuration makes use of a low percentage of CPU. To evaluate the CPU load we exploited the Linux *Top* program, which provides a dynamic real-time view of the running processes and system's resources usage. Results show that both ntpd slaves and PTP daemons (ptp4l and phc2sys) require a negligible overhead (*i.e.*, close to 0 %).

## 4.4 How-To Guide - NTP / PTP Settings



NTP / PTP software settings

| Daemon | Masters | Slaves |
|---|---|---|
| ntpd | - Synch to pool of servers (Default 1024 s):<br>pool IP-S0 iburst<br>pool IP-Sn iburst<br>- Set Node as LAN Master:<br>server 127.127.1.0<br>fudge 127.127.1.0 stratum 1 | - Synch to LAN Masters (rate 8 s):<br>server IP-M0 maxpoll 3<br>server IP-Mn maxpoll 3 |
| ptp4l | - Default settings (rate 1 s) | - Default settings (rate 1 s) |
| phc2sys | - Rate 12 Hz<br>- use -w, to wait until ptp4l is synchronized | - Rate 12 Hz<br>- use -w, to wait until ptp4l is synchronized |

**Figure 8: Proposed network and software configuration.**

In this final section we report a short "how-to" guide, based on the experience gained in this work to achieve the presented synchronization performance. Figure 8 (left) shows the network configuration with the respective polling rates of NTP and PTP. As in the target scenario (HPC systems) is only important that all devices within the LAN are highly synchronized between them, and we can accept a lower accuracy to the absolute time (external world), we use few internal NTP servers that synchronize their system clock to a pool of external NTP servers (standard configuration of NTP) with the default polling rate of 1024 s. It must be noted that we use more than one internal server synchronized to the external world to achieve a more resilient configuration. All slave devices within the LAN are then synchronized with NTP polling rate of 8 s or PTP rate of 1 s. Figure 8 (right) summarizes our software settings.

# 5 CONCLUSION

High synchronization performance is required to address several key challenges in today and future leadership-class computing centers. In this paper we evaluated the two widely used network synchronization protocols, NTP and PTP, in a SoA HPC Cluster, namely D.A.V.I.D.E.. Our results show that NTP can reach an accuracy of 2.6 μs and a precision below 2.7 μs on computing nodes, while maintaining a negligible overhead. Such values can be bounded to sub-microseconds when using PTP and low-cost COTS switches. We then tested these protocols on embedded devices (*i.e.*, BBB) which are part of a SoA (best-in-class) out-of-band power and performance monitoring infrastructures, namely DiG, and results show they are also suitable for data timestamping and correlation. Finally, we quantified their scalability and CPU overhead. In light of the obtained results, we can state that both protocols can provide the target synchronization of sub-milliseconds needed for many potential applications in today's supercomputers.

# ACKNOWLEDGMENTS

This work is partially funded by the EC under program grant H2020 FET-HPC ANTAREX 671623 and by E4 Computer Engineering SpA. The authors would like to thank the Italian supercomputing center CINECA and all the team of E4 Computer Engineering SpA for their valuable help and support.

# REFERENCES

[1] 2008. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.* https://doi.org/10.1109/IEEESTD.2008.4579760

[2] W. Abu Ahmad et al. 2017. Design of an Energy Aware Petaflops Class High Performance Cluster Based on Power Architecture. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 964–973. https://doi.org/10.1109/IPDPSW.2017.22

[3] A. Bartolini et al. 2018. The D.A.V.I.D.E. Big-Data-Powered Fine-Grain Power and Performance Monitoring Support. In *ACM International Conference on Computing Frontiers 2018.* https://doi.org/10.1145/3203217.3205863

[4] D. Becker et al. 2008. Implications of non-constant clock drifts for the timestamps of concurrent events. In *2008 IEEE International Conference on Cluster Computing.* 59–68. https://doi.org/10.1109/CLUSTR.2008.4663756

[5] D. Becker et al. 2009. Scalable timestamp synchronization for event traces of message-passing applications. *Parallel Comput.* 35, 12 (2009), 595 – 607. https://doi.org/10.1016/j.parco.2008.12.012 Selected papers from the 14th European PVM/MPI Users Group Meeting.

[6] D. Becker et al. 2010. Synchronizing the Timestamps of Concurrent Events in Traces of Hybrid MPI/OpenMP Applications. In *2010 IEEE International Conference on Cluster Computing.* https://doi.org/10.1109/CLUSTER.2010.13

[7] J. Burbank et al. 2015. Network Time Protocol Version 4: Protocol and Algorithms Specification. IETF RFC 5905. https://doi.org/10.17487/rfc5905

[8] CISCO Systems, Inc. 2016. *Cisco Industrial Ethernet 3000 Layer 2/Layer 3 Series Switches.* Datasheet.

[9] C. Conficoni et al. 2015. Energy-aware cooling for hot-water cooled supercomputers. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE).* 1353–1358.

[10] B. Ferencz et al. 2013. Hardware assisted COTS IEEE 1588 solution for x86 Linux and its performance evaluation. In *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings.* https://doi.org/10.1109/ISPCS.2013.6644762

[11] P. Giannozzi et al. 2009. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter* 21, 39 (2009), 395502.

[12] T. Ilsche et al. 2018. Power measurement techniques for energy-efficient computing: reconciling scalability, resolution, and accuracy. *Computer Science - Research and Development* (Apr 2018). https://doi.org/10.1007/s00450-018-0392-9

[13] T. Jones et al. 2017. An evaluation of the state of time synchronization on leadership class supercomputers. *Concurrency and Computation: Practice and Experience* 30, 4 (2017), e4341. https://doi.org/10.1002/cpe.4341 e4341 cpe.4341.

[14] T. Jones and G. A. Koenig. 2012. Clock synchronization in high-end computing environments: a strategy for minimizing clock variance at runtime. *Concurrency and Computation: Practice and Experience* 25, 6 (2012), 881–897. https://doi.org/10.1002/cpe.2868

[15] A. Libri et al. 2016. Evaluation of synchronization protocols for fine-grain HPC sensor data time-stamping and collection. In *2016 International Conference on High Performance Computing Simulation (HPCS).* 818–825. https://doi.org/10.1109/HPCSim.2016.7568419

[16] A. Libri et al. 2018. Dwarf in a Giant: Enabling Scalable, High-Resolution HPC Energy Monitoring for Real-Time Profiling and Analytics. *ArXiv e-prints* (June 2018). arXiv:cs.DC/1806.02698

[17] Mellanox Technologies. 2015. *EDR InfiniBand.* OFA UM 2015, OpenFabrics Software User Group Workshop.

[18] J. Ridoux and D. Veitch. 2009. Ten Microseconds Over LAN, for Free (Extended). *IEEE Transactions on Instrumentation and Measurement* 58, 6 (June 2009), 1841–1848. https://doi.org/10.1109/TIM.2009.2013653

[19] J. Serrano et al. 2013. THE WHITE RABBIT PROJECT. In *Proceedings of IBIC2013, Oxford, UK.* http://cds.cern.ch/record/1743073

[20] C. Silvano et al. 2017. The ANTAREX Tool Flow for Monitoring and Autotuning Energy Efficient HPC Systems. In *SAMOS 2017 - International Conference on Embedded Computer Systems: Architecture, Modeling and Simulation.* Pythagorion, Greece. https://hal.inria.fr/hal-01615945