Work-in-Progress: A Novel Clock Synchronization System for Large-Scale Clusters

Zhuochen Fan*†, Xiaodong Li*†, Yanwei Xu†, Yuqing Li‡, Tong Yang*, Steve Uhlig§
*School of Computer Science, and National Engineering Laboratory for Big Data Analysis Technology and Application, Peking University, China †Theory Lab, Central Research Institute, 2012 Labs, Huawei Technologies Co., Ltd., China
‡School of Cyber Science and Engineering, Wuhan University, China
§School of Electronic Engineering and Computer Science, Queen Mary University of London, UK

***Abstract*—Clock synchronization is essential in real-time applications of large-scale clusters. State-of-the-art Huygens clock synchronization reduces synchronization errors through offset probing loop correction between data center servers. However, Huygens does not offer a solution for large-scale clusters. In this paper, we propose a novel and scalable `CAT-Sync` clock synchronization system for large-scale clusters, which includes three key techniques: optimal probe topology Construction, probing channel Assignment, and Time-slice synchronization. In `CAT-Sync`, the workload of each host is the same and will not increase with the expansion of the cluster size. Our `CAT-Sync` system achieves a stable clock synchronization accuracy within 2 microseconds on 60 virtual machines, and the average clock offset for the entire synchronization process is improved by about 44.8% compared to Huygens.**

***Index Terms*—clock synchronization, cluster, probe, offset, topology, edge coloring, broadcast, time slice**

I. Introduction

A. Background and Motivations

With the development of high-performance computing technology and services, the scale of computer systems needed to handle existing workloads continues to expand. Many distributed applications nowadays require the use of networks of clusters of host devices [1]. Clock synchronization is essential in clustered system applications, providing support for functions such as data sharing, coordination between devices, and interaction at specified times.

Large-scale cluster clock synchronization is based on precise synchronization between the clocks of two hosts. The mainstream clock synchronization schemes, including the widely used NTP [2], PTP [3], DTP [4], Huygens [5], *etc.*, send probes in the form of timestamped packets. The key idea of packet probing is to determine the offset between two clocks by estimating the one-way delay (OWD). Specifically, when Host A needs to synchronize Host B's clock, Host A first sends a probe packet 1 to Host B, and records the sending time TX_A. When Host B recognizes probe packet 1, it records its receiving time RX_B, sends a probe packet 2 to Host A, and returns RX_B and the sending time of probe packet 2 tx_B. Host A records its receiving time rx_A after receiving probe packet 2. The probe host calculates OWD and offset from the four timestamps mentioned above. If the local real-world clock time is t, the offset of Host A and Host B relative to t is Δ_A and Δ_B, then A's clock time is $t_A = t + \Delta_A$, B's clock time is $t_B = t + \Delta_B$, and the offset between hosts is $\Delta_{AB} = \Delta_B - \Delta_A$. Under the assumption that the OWD of probe packet 1 and probe packet 2 are the same, Host A can calculate the OWD $O = \frac{(RX_B - TX_A) + (rx_A - tx_B)}{2}$ and offset $\Delta_{AB} = \frac{(RX_B - TX_A) - (tx_B - rx_A)}{2}$.

Based on O and Δ_{AB}, the probe host can modify the local clock to synchronize with the probed host. Existing clock synchronization schemes are based on these four timestamps, which differ in how timestamps are generated and processed.

B. Prior Art and Limitations

Huygens [5] is a high-precision clock synchronization scheme for data center networks, which reduces synchronization errors through *Loop Correction* for offset probing between servers (see the next paragraph for details). It uses SVM to batch process all probe packets between any two hosts in a certain time interval, to filter out path noise and obtain the offset between the two clocks. Thus, the main factor affecting the accuracy of offset measurement is whether the round-trip path is *symmetrical*, *e.g.*, the number of switch hops passed by the round-trip probing path between nodes may be different.

Since clock synchronization is symmetrical and transitive, *i.e.*, $\Delta_{AB} = \Delta_B - \Delta_A = -\Delta_{BA}$, $\Delta_{AC} = \Delta_{AB} + \Delta_{BC}$, ideally the sum of the loop clock offset is 0. However, due to measurement errors, the measured loop offset sum is often different from 0. To ensure optimal synchronization performance, Huygens uses the minimum norm solution to distribute the offset errors of the loop to different paths. Specifically, given the probing topology $G(V, E)$, the corresponding loop matrix $A_{|L|\times|E|}$ are obtained, where each column corresponds to a directed edge $(i, j) \in E$ of G[1], and each row corresponds to a linear independent loop of $l_n \in L$. If $(i \to j) \in l_n$ is a forward edge on E, the corresponding element of A is 1, the reverse edge is -1, and the other is 0. The current offset probe

The first two authors contribute equally. Corresponding authors: Yanwei Xu (xuyanwei1@huawei.com) and Tong Yang (yangtongemail@gmail.com). This work is supported by National Key R&D Program of China (2022YFB2901504), and National Natural Science Foundation of China (NSFC) (No. U20A20179).

[1] i and j represent two servers in Huygens, and two hosts in this paper.

2022 IEEE Real-Time Systems Symposium (RTSS) | 978-1-6654-5346-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/RTSS55097.2022.00056

2576-3172/22/$31.00 ©2022 IEEE
DOI 10.1109/RTSS55097.2022.00056

between all host clocks is denoted as $M = [M_e]_{|E|\times 1}$. The loop error $Y = [y_l]_{|L|\times 1}$ is obtained through $AM = Y$. The error y_l of each loop is assigned to the contained probe edge via $\Delta M = A^T(AA^T)^{-1}Y$, where $\Delta M = [||\Delta M_e||]_{|E|\times 1}$, and the offset correction corresponding to each edge $e \in E$ is ΔM_e. Finally, the error compensation is performed on the results, $i.e.$, the offset optimized for the loop is $M - \Delta M$.

However, Huygens suffers from the following limitations that prevent it from being deployed in large-scale clusters: 1) Huygens does not specify how to build a probe topology; 2) Huygens does not provide how to coordinate globally across different probe sessions to avoid overwrite errors when a large number of timestamps interact at the same time; 3) When the clocks in the cluster are not synchronized, It is difficult for Huygens to ensure that all clock gaps between the current hosts are measured at the same time.

C. Our Solution

Based on the above analysis, we propose a novel clock synchronization system named `CAT-Sync` for large-scale clusters, designed to address the following challenges:

1) **Probe Topology Building.** Clock synchronization in a cluster requires synchronizing all hosts to one of the reference hosts. Since there is an upper limit to the number of probed sessions that each host can support, it is necessary to synchronize the other hosts to the reference clock in a hierarchical manner. However, the error increases with the layers of probe relationships. Also, the effects of different loops on error optimization is different under different probe topologies. *To address this, we propose an optimal probe topology **C**onstruction scheme based on graph theory, to achieve the lowest possible resource consumption.* See Section II-B for details.

2) **Probe Session Conflict.** The generation of hardware timestamps relies on the time register in the network card, which usually has only one register due to the cost limitation. Thus, if the clock probe packet on the host does not read the timestamp data into the system before the next packet arrives, an overwrite error will occur. According to our tests, it takes more than $100\mu s$ between the arrival of two probe packets. *To address this, we propose to exploit the edge coloring of graphs to realize global time-division channel **A**ssignment of probe sessions.* See Section II-C for details.

3) **Probe Session Synchronization.** The probe loop optimizes the error on the premise that the clock gaps between the hosts are measured at the same time. This is difficult to guarantee if the synchronization between the clocks is not reached. *To address this, we propose a probing **T**ime-slice clock synchronization scheme based on message propagation tree and controller broadcasting.* See Section II-D for details.

II. System Design

A. Overall

The workflow of `CAT-Sync` is as follows. The cluster controller is responsible for the management and control of the entire clock synchronization cluster. Once the controller is started, a listening port is opened to accept incoming connections from the host node. Upon receiving a connection request, `CAT-Sync` creates a separate persistent TCP connection for each host (for data transmission) and a regular heartbeat thread to verify that the connection is still working properly. Session management information can be obtained in real-time, e.g., hosts joining and leaving. This allows us to keep the probe topology up to date. For a synchronous cluster composed of all connected host nodes, the controller constructs the optimal probe topology under the limitation of single-machine probe sessions, and uses the edge coloring algorithm of the graph to globally allocate the time probe channels. In each time slice, according to the assigned probe tasks and session time-division channels, hosts send probe packets to each other through the connection and obtain the timestamps of the probe packets. Controller broadcasts and message tree-based messaging guarantee a globally synchronized start and end of the current time slice. At the end of each time slice, the probe host collects the corresponding timestamp data for all probe sessions in the current time slice, calculates the clock drift and offset with all the probed hosts, and reports them to the controller. After receiving the probe results between all hosts in the current time slice, the controller applies probe loop optimization to correct errors. Then, it calculates the drift of each host relative to the reference clock and the offset at the end of the current time slice, and sends them to all hosts. Finally, the host adjusts its local clock based on the received optimized probe results.

B. Optimal Probe Topology Construction

Once the probe session management is performed on the currently connected host node, the controller constructs the optimal probe topology and maximizes the use of the probe loop to optimize synchronization accuracy.

Since there is an upper limit on the number of probing sessions that each host can support, the cluster probing topology optimization problem is equivalent to: given the number of nodes N and the maximum node degree K, we construct an undirected connected graph $G(V, E)$ such that each node has at least in a loop $l_n \in L$, where edge $(i, j) \in E$ indicates that there is a probing relationship between two nodes. Hence, our construction must **trade off** the following two aspects:

1) The greater the distance d from the host $v \in V$ to the reference clock, the more the accuracy drops. Therefore, it should be synchronized to the reference clock along the shortest path $|d_v|$;
2) The larger the cumulative sum of linear independent loop lengths $\sum_{l_n \in L} |l_n|$, the more non-zero entries in the probing loop matrix A, and the smaller the loop correction error $||\Delta M_e||$ based on the minimum norm solution.

1) To achieve the minimum synchronization distance between nodes, we construct a hierarchical topology based on the BFS spanning tree with the reference clock C_0 as the root node. We assume that each node is no more than d hops from C_0. Except for leaf nodes, the node degree should

be K, $i.e.$, each node in d-th layer is connected to one node in the $(d-1)$-th layer, and is connected to $K-1$ nodes in the $(d+1)$-th layer. Therefore, the total number of nodes in the d-th layer is $N_d = K(K-1)^{d-1}$.

2) To maximize the synchronization accuracy of loop correction, we add edges based on the above hierarchical spanning tree topology, to form a set of linear independent loops with the maximum cumulative sum of loop lengths. For an undirected connected graph $G(V, E)$, the set of linear independent loops ($i.e.$, the basis of the linear equation system) based on the spanning tree T is uniquely determined, where each independent loop corresponds to a closed loop $(f, g, \ldots, f)$ consisting of any edge (f, g) on G-T and the upper path $(g, \ldots, f)$ on T, respectively. Since the number of nodes, edges, and loops satisfy the constraint $|L| = |E| - |V| + 1$, $i.e.$, given the number of nodes $|V|$, more edges provide more linearly independent loops. Therefore, the optimal probe topology construction problem is equivalent to how to add $M = \frac{KN}{2} - N + 1$ edges to the last layer to make it K-edges connected, so that the loop length is the maximum value of $2d+1$. We propose a feasible solution of calculating the set of feasible leaf nodes and the shortest path between any two leaf nodes, and then sort the feasible edges based on the shortest path and add edges in turn.

Remark (Figure 1). For the operation of adding edges to leaf nodes to form probe loops described by **2)**, it can be divided into the following two cases:

Case 1: When the last layer of leaf nodes are full, $i.e.$, when the number of nodes satisfies $N = 1 + \frac{K\left(1-(K-1)^d\right)}{K-2}$, the scheme of adding edges to the spanning tree is that the leaf nodes are connected to the leaf nodes with an integer multiple of $K-1$ in turn. In Figure 1, node 8 is connected to nodes 5, 11, and 14.

Case 2: When the last layer of leaf nodes is dissatisfied: $i.e.$, when some leaf nodes of the d-th layer do not exist, each $K-1$ edges that should connect them respectively is processed as follows: 1) We arbitrarily select an edge and connect it to its parent node; 2) We connect the remaining $K-2$ edges each other between different sub-trees of C_0 according to the two vertices of each edge. Assuming that the leaf node 16 in Figure 1 does not exist, then the nodes 7, 10, and 13 that should be connected to it are processed as: node 7 is connected to its parent node 4, and nodes 10 and 13 are directly connected.

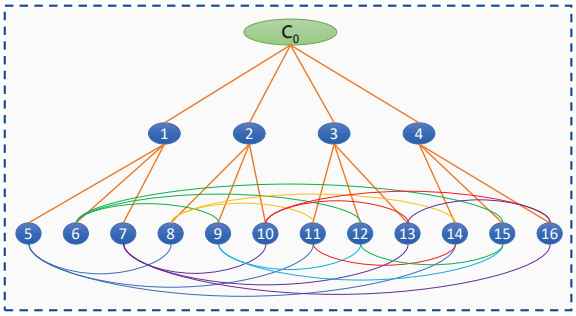

Fig. 1. An example of optimal topology construction: $K = 4$.

C. Probing Channel Assignment

We choose a time channel for the probe session or probe edge between hosts in the cluster. This is necessary to allow the timestamp data of the current packet on each host to be read before the next packet arrives, avoiding overwriting when a large number of timestamps interact. It consists of the following two steps: 1) *We utilize the Misa-Gries coloring algorithm [6] to color the edges of the probe topology (V, E) so that any two edges are not the same color.* In this way, we can find an edge coloring scheme for any graph in polynomial time, and the number of colors required is no more than $K+1$, where K is the maximum node degree; 2) *We perform a global assignment of probing session channels according to the obtained edge coloring scheme, and assign one color of an edge to a unique time channel, as shown in Figure 2.* Since at most $K+1$ channels are needed to complete the channel allocation, a static channel allocation scheme can be generated to ensure the efficiency of the coloring algorithm. When the topology changes, such as when a host exits or a new host joins, the channel allocation scheme of the probe session is updated.

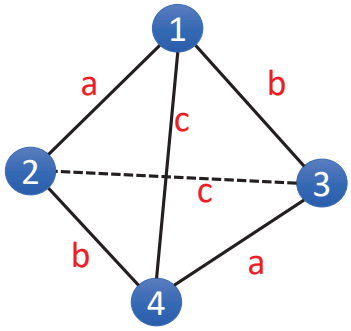

Fig. 2. An example of assigning time-division channels to each probing session, where $K = 3$. It requires only 3 colors in total (for a, b, c).

D. Time-Slice Synchronization

Time-slice synchronization includes the simultaneous start and simultaneous end of time slices. Simultaneous starts serve the purpose of keeping the probing time channels aligned on all hosts. Simultaneous ends deal with loop correction probing errors. The precise synchronization of time slices among hosts on the entire cluster depends on sending time-slice end messages between the hosts and between the hosts and controller. For this, the message only needs to include the time slice ID. The message is forwarded and processed as the highest priority message within the hosts and controller. To avoid creating a storm of messages, the hosts or controller should block the processing of subsequent messages with the same ID after receiving an end message with a time slice ID. It includes the following three steps: 1) *We build a BFS spanning tree based on the probe topology with each host node as the root, and obtain the propagation path of the time-slice synchronization message starting at that node as the spanning tree rooted with it for any host*; 2) In each time slice: On the sending side, the host judges whether the current time slice ends according to the local clock. Once it

ends, it immediately stops sending and receiving local probe packets, and simultaneously sends time-slice end messages to the controller and neighboring hosts along the message propagation path; On the listening side, the host listens for time-slice end messages from controller or other hosts. Once received, it immediately stops the local probing, and forwards time-slice end messages to all neighboring hosts; 3) *After the controller receives the first time-slice end message of the host, it generates a broadcast message to all host nodes, and synchronizes the entire hosts to enter a new time-slice.*

III. Evaluation

Our `CAT-Sync` *is* ***scalable*** *to support clusters of arbitrary size. Since the number of probes for each host is fixed, the workload of each node is fixed. Therefore, large-scale cluster deployment with any number of nodes can be supported.*

A. Experimental Setup

Implementation. We build a cluster consisting of 60 virtual machines for the experimental evaluation of `CAT-Sync`, where the probe interval is 20000μs and the time-slice length is 10s. We construct a physical star topology of 60 virtual machines and an OVS switch. The number of channels is 7, so for this topology: $K = 6$, $d = 4$ (see Section II-B). Based on this physical network topology, we construct experiments on `CAT-Sync` and Huygens logical topologies, respectively.

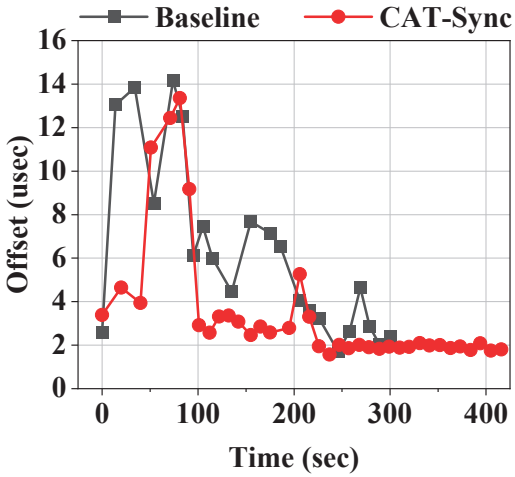

Fig. 3. Clock offset for long-term synchronization.

Experimental platform. Our experiments are performed on the Klonet network simulation experiment platform [7], whose architecture includes: 1) infrastructure, 2) simulated network, 3) control framework, and 4) application interface[2].

Baseline. We use the original **Huygens** [5] as the baseline solution. We first select one machine as the controller device and reference clock, and build loop topology around this reference clock, with a maximum of 20 nodes in each loop. In this experiment, we build 3 loops around the reference clock.

[2]1) The infrastructure is the foundation of the platform network and computing resources; 2) The simulated network is the virtual network deployed by the platform; 3) The control framework is a collection of various control logics of the platform, responsible for parsing instructions and executing various operations; 4) The application interface is the interface for users to interact with the platform and conduct experiments.

B. Experimental Results

From Figure 3, the experimental results show that the clock offset of `CAT-Sync` has obvious advantages compared to Huygens: While the offset of `CAT-Sync` has been relatively stable around 100 seconds, the offset of Huygens does not gradually stabilize until beyond 300 seconds. Finally, the offset of `CAT-Sync` is absolutely stable within 2μs, while Huygens is still in large fluctuations. During the entire synchronization process, the average offset of `CAT-Sync` is 3.48μs, which is 55.8% times that of Huygens. In other words, `CAT-Sync` achieves 44.2% improvement over Huygens. In Figure 4, we find the drift distribution of `CAT-Sync` is significantly closer to the y-axis, which also demonstrates that its offset is better.

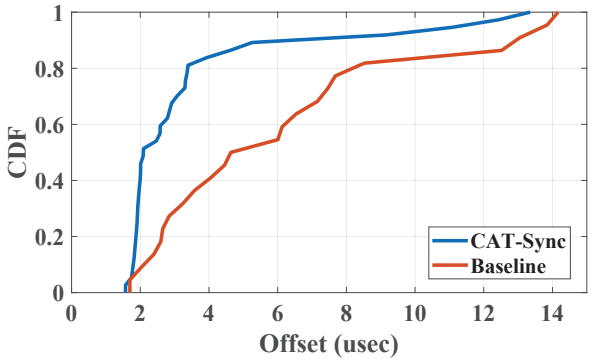

Fig. 4. The CDF of `CAT-Sync` and Baseline on clock offset.

IV. Conclusion and Future Work

This paper analyzes the limitations of the state-of-the-art Huygens scheme and proposes a novel and scalable `CAT-Sync` synchronization system specially designed for large-scale clusters. `CAT-Sync` includes 3 key techniques and achieves a stable clock offset of less than 2μs in the clusters composed of 60 virtual machines. As future work, we plan to further optimize the proposed 3 key techniques, expand the cluster size for future experiments, and conduct full testbed implementations on more real-world platforms.

References

[1] N. Shivaraman, P. Schuster, S. Ramanathan, A. Easwaran, and S. Steinhorst, "Cluster-based network time synchronization for resilience with energy efficiency," in *Proc. RTSS*, 2021, pp. 149–161.

[2] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans Commun*, vol. 39, no. 10, pp. 1482–1493, 1991.

[3] J. C. Eidson, M. Fischer, and J. White, "Ieee-1588™ standard for a precision clock synchronization protocol for networked measurement and control systems," in *Proc. PTTI*, 2002, pp. 243–254.

[4] K. S. Lee, H. Wang, V. Shrivastav, and H. Weatherspoon, "Globally synchronized time via datacenter networks," in *Proc. SIGCOMM*, 2016, pp. 454–467.

[5] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *Proc. NSDI*, 2018, pp. 81–94.

[6] J. Misra and D. Gries, "Finding repeated elements," *Sci Comput Program*, vol. 2, no. 2, pp. 143–152, 1982.

[7] J. Xie, W. Shan, C. Xiao, T. Ma, L. Chen, H. Yu, and G. Sun, "Klonet: a network emulation platform for the technology innovation," *Telecommunications Science*, vol. 37, no. 10, pp. 66–75, 2021.