A PROOFS

A.1 Proof of Proposition 4.1

Proof. For $G = FP$ (or $G = LP$), as defined in Section 4.1, P_G is the point with both the minimum time (or the maximum time) and the largest version number. Therefore, no other chunk in $\mathbb{C}''$ with a version number larger than $P_G.\kappa$ contains a point with the same time as P_G. In other words, P_G is never updated by later appended chunks, having $P_G.t \notin C^{\kappa_1}$ for any $C^{\kappa_1} \in \mathbb{C}''$ with $P_G.\kappa < \kappa_1$. Moreover, P_G is not covered by the delete time range of any delete in $\mathbb{D}$ with a larger version number than $P_G.\kappa$, i.e., $P_G.t \notin D^{\kappa}, D^{\kappa} \in \mathbb{D}, \kappa > P_G.\kappa$. Referring to Formula 2 in Definition 2.7, P_G is a point in the merged time series $M(\mathbb{C}'', \mathbb{D})$, i.e., the latest.

Next we prove that the latest candidate point is the representation result for $G \in \{FP, LP\}$. From Definition 2.7, we know that if P_G is the latest, P_G is a point in $T_i = M(\mathbb{C}'', \mathbb{D})$. Therefore, T_i can be divided into three disjoint subsequences based on P_G,

$$T_l = \{P \mid P \in T_i, P.t < P_G.t\},$$
$$T_m = \{P_G\},$$
$$T_r = \{P \mid P \in T_i, P.t > P_G.t\}.$$

From the distributive property [20] of G, we have

$$G(T_i) = G(T_l \cup T_m \cup T_r) = G(\{G(T_l), G(T_m), G(T_r)\}) = G(\{G(T_l), P_G, G(T_r)\}).$$

It is easy to know that the chunk metadata bounds the time and value range of all points in the chunk. Since P_G is extracted from the boundary points among all chunk metadata, we have

$$P_G.t \leq G(T_i).t \leq \min(G(T_l).t, G(T_r).t), G = FP$$
$$P_G.t \geq G(T_i).t \geq \max(G(T_l).t, G(T_r).t), G = LP$$

i.e., $G(T_i) = G(\{G(T_l), P_G, G(T_r)\}) = P_G$, for $G \in \{FP, LP\}$. □

A.2 Proof of Proposition 4.3

Proof. The first condition states that P_G is not updated by any later appended chunks, having $P_G.t \notin C^{\kappa}$ for any $C^{\kappa} \in \mathbb{C}''$ with $\kappa > P_G.\kappa$. The second condition states that P_G is not covered by any delete in $\mathbb{D}$ with a larger version number than $P_G.\kappa$, i.e., $P_G.t \notin D^{\kappa}$ for any $D^{\kappa} \in \mathbb{D}$ with $\kappa > P_G.\kappa$. Referring to Formula 2 in Definition 2.7, P_G is a point in the merged time series $M(\mathbb{C}'', \mathbb{D})$, i.e., the latest. With the latest candidate point, the proof of the representation result follows the same line of Proposition 4.1. □

B USE CASES

Steel Manufacturing. In a steel manufacturer, Apache IoTDB manages the time series of 250,000 sensors in 30,000 devices. The data collection interval ranges from 10 milliseconds to 5 seconds, generating 1.5 billion points every day. A dashboard is developed based on our proposed solution, to visualize the temperatures at different stages of the steel manufacturing process, including blast furnace molten iron temperature, molten steel temperature, refining entry temperature, refining exit temperature, tunnel temperature, and so on. Domain experts inspect the visualized time series of temperatures to explore the potential gaps among different stages. By optimizing the steel manufacturing process, such temperature gaps are expected to be reduced to improve energy efficiency.

Aviation Industry. Apache IoTDB is employed by a company in the aviation industry to store the high frequency device test data at nanosecond level. Using the IEEE 1588 Precision Time Protocol (PTP), the data collection frequency is as high as 20kHz to 400kHz, generating more than 1T time series in each test task. Our proposed solution is used to visualize and compare the overall performance metrics with those of each part. The domain experts can thus evaluate whether the changed design improves the overall/part performances.

Cloud Service. Apache IoTDB is used as the time series database in an Application Performance Monitoring (APM) product. It monitors multiple metrics of the hosts in every 20 milliseconds to 10 seconds, including CPU and network utilization, GVM and heap memory usage, etc. The time series collected by a single metric in one day can reach 4.32 million points. Our solution is employed to visualize multiple metrics on the same dashboard for fault diagnosis of application performance. For example, by comparing the garbage collection time and the heap usage trend, users can analyze memory leaks.