# B ADDITIONAL MATERIAL

## B.1 Generalization to Other Models

We start with the simple time series forecasting models AR in this study. The proposed methods can be easily generalized to more common AR variants, such as MA and ARMA. We pre-compute the auto-covariance as stated in Formula (1), while applying different solving methods to estimate the model coefficients. For MA models, there is a bijection between the MA parameters and auto-covariances, referring to [Tsay 2005]. Thus we can also use the aggregated auto-covariances in LSM storage to speed up estimation. For estimating ARMA parameters, we need to first estimate AR parameters, use the estimated AR to calculate the residual error at each timestamp, and then estimate the MA parameters on the residuals. The pre-computed auto-covariance accelerates the part of estimating AR models for ARMA. Figure 12 reports the time cost of AR, MA, ARMA and their corresponding LSM deployments. As illustrated, similar to AR models, the learning performances of both MA and ARMA are significantly improved by our proposals.

[Tsay 2005] Tsay, R. S: Analysis of financial time series. John wiley & sons 2005.

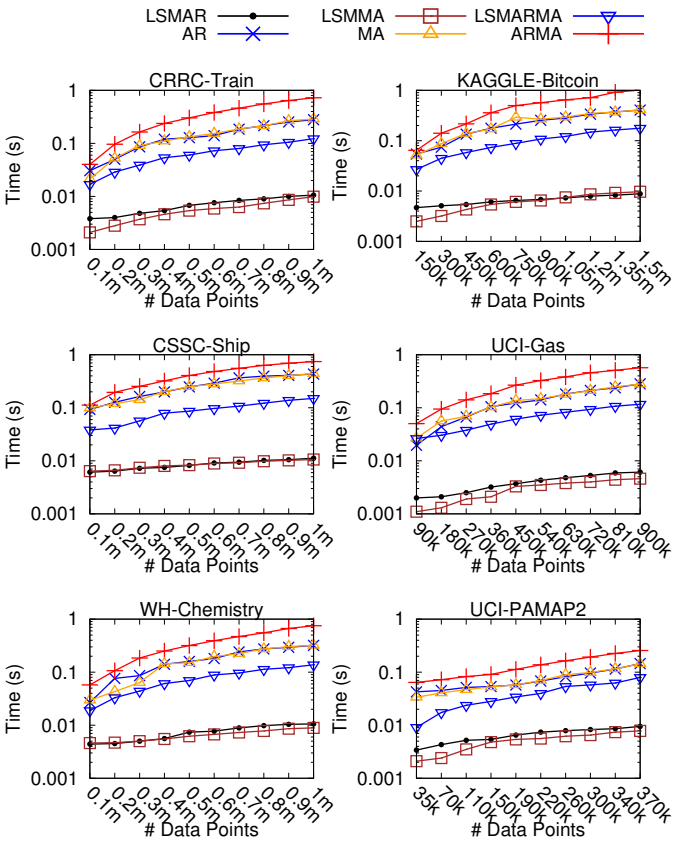


**Figure 12: Improvement on performances of learning various models**

## B.2 Comparison to Other Learning Methods

We evaluate three state of the art time series analysis methods based on representation learning, including Transformer [Vaswani et al. NIPS 2017], Informer [Zhou et al. AAAI 2021], Pyraformer [Liu et al. ICLR 2022]. It is still an open problem on how to optimize such representation learning in an LSM-tree based time series store. Nevertheless, we report a comparison with these methods on learning accuracy and time cost in Table 4 below. It is not surprising that the AR models have higher MSE than representation learning based methods (comparable in some datasets such as CSSC-Ship and UCI-PAMAP2). The learning time performance of our LSMAR is at least 4 orders of magnitude faster. For the applications of end devices in the IoT scenarios, where Apache IoTDB can still be installed, our efficient LSMAR applies while the deep learning based solutions are too heavy to perform.

[Vaswani et al. NIPS 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: Attention is All you Need. NIPS 2017: 5998-6008

[Zhou et al. AAAI 2021] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, Wancai Zhang: Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. AAAI 2021: 11106-11115

[Liu et al. ICLR 2022] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, Schahram Dustdar: Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. ICLR 2022

## B.3 Comparison to Other Accelerating Method

We evaluate the most related work on the learning strategy for decentralized parallel nodes [Lian et al. ICML 2018]. It averages the local model of each node (e.g., in page 1 in Figure 1) with the model of its neighbor (i.e., in page 2). The imputed, repaired, delayed, and re-transmitted data between two pages are ignored. Thereby, the learned models may not be as accurate as those by considering the imputed, repaired, delayed, and re-transmitted data. The results are reported in Figures 13 and 14. Without considering the imputed, repaired, delayed, or re-transmitted data, the decentralized learning has higher MSE, but lower time cost. Nevertheless, even for non-overlapping pages without imputed, repaired, delayed, or re-transmitted data, our proposal is still more accurate, since the decentralized learning approximates the models by simply averaging among pages.

[Lian et al. ICML 2018] Xiangru Lian, Wei Zhang, Ce Zhang, Ji Liu: Asynchronous Decentralized Parallel Stochastic Gradient Descent. ICML 2018: 3049-3058

## B.4 Implementation in Other Platforms

TimescaleDB is built upon a relational database, where imputed, repaired, delayed, or re-transmitted data are inserted in place. Thereby, it corresponds to the special cases of adjacent and disjoint pages in Sections 4.1 and 4.2, without overlapping pages. In this sense, our proposal can also be applied to TimescaleDB. We are now working on the implementation and the results will be added soon. The results are expected to be similar to Figure 10 under different disjoint length, without overlapping pages.

| Dataset | LSMAR | | Transformer | | Informer | | Pyraformer | |
|---|---|---|---|---|---|---|---|---|
| | Time | MSE | Time | MSE | Time | MSE | Time | MSE |
| CRRC-Train | 0.011 | 1.76 | 653 | 0.099 | 601 | 0.086 | 650 | 0.080 |
| CSSC-Ship | 0.011 | 0.085 | 674 | 0.076 | 617 | 0.080 | 657 | 0.080 |
| WH-Chemistry | 0.010 | 0.026 | 493 | 0.020 | 600 | 0.0084 | 627 | 0.0073 |
| KAGGLE-Bitcoin | 0.012 | 3.61 | 665 | 1.42 | 621 | 1.26 | 664 | 1.28 |
| UCI-Gas | 0.010 | 1.35 | 623 | 0.32 | 591 | 0.12 | 639 | 0.13 |
| UCI-PAMAP2 | 0.0095 | 0.99 | 177 | 0.97 | 212 | 0.96 | 233 | 0.98 |

**Table 4: Learning time costs and MSE of different learning methods**

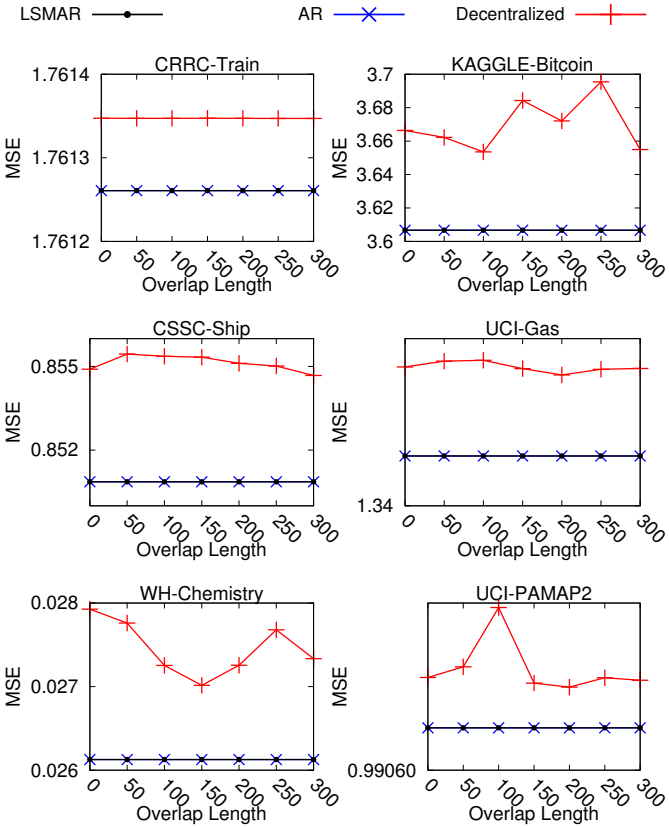


**Figure 13: MSE of different accelerating learning methods**

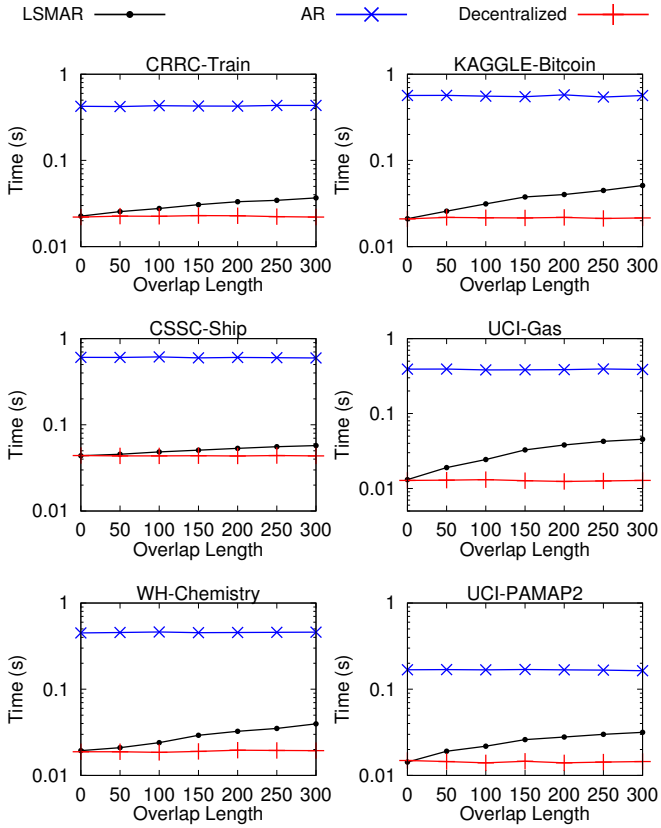


**Figure 14: Time cost of different accelerating learning methods**