

# Fast checkpointing of Large Language Models with TensorStore CHFS

SOHEI KOYAMA, KOHEI HIRAGA (ADVISOR), and OSAMU TATEBE (ADVISOR), University of Tsukuba

The frequency of checkpoint creation in large language models is limited by the write bandwidth to a parallel file system. In this study, we aim to reduce the checkpoint creation time by writing to the Intel Optane Persistent Memory installed on the compute nodes. We propose TensorStore CHFS, a storage driver that adds an ad hoc parallel file system CHFS to the TensorStore. The proposed method succeeded in increasing the checkpoint creation bandwidth of the T5 1.1 model by 4.5 times on 32 nodes.

CCS Concepts: • **Computing methodologies** → *Distributed computing methodologies*;

Additional Key Words and Phrases: large language models, distributed training, checkpointing

## ACM Reference format:

Sohei Koyama, Kohei Hiraga (advisor), and Osamu Tatebe (advisor). 2023. Fast checkpointing of Large Language Models with TensorStore CHFS. 1, 1, Article 1 (August 2023), 2 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

Large language models have attracted the attention of industry and research community because of their utility [7]. Training large language models requires enormous computational resources, and state preservation during model training, which is also known as checkpointing, is time- and resource-intensive. Although creating checkpoints at a high frequency is desirable, the frequency of checkpoints is limited by the write bandwidth to the parallel file system [2]. In this study, we propose the TensorStore consistent hashing file system (CHFS) to accelerate checkpoint creation in training large language models. CHFS [6], an ad hoc parallel file system, is constructed by bundling the persistent memory of compute nodes. Furthermore, checkpoints are created on the CHFS to accelerate checkpoint creation. In this study, the CHFS was integrated into TensorStore [3] so that it could be used seamlessly from T5X [5], Orbx [4], and JAX [1]. The evaluation results revealed that the bandwidth of checkpoint creation can be improved by up to ~4.5 times using the CHFS.

## 2 RELATED RESEARCH

Check-N-Run [2] is a scalable checkpoint system used to train large-scale recommendation models. It uses incremental checkpointing and quantization techniques to reduce data size without compromising accuracy. Our study did not reduce the data size, but improved the storage performance, accelerating checkpoint creation.

## 3 BACKGROUND

### 3.1 TensorStore

TensorStore [3] is a library designed for efficient reading and writing of large-scale multidimensional arrays. The Listing 1 illustrates the method of using TensorStore.

```
import tensorstore as ts
dataset = ts.open({
```

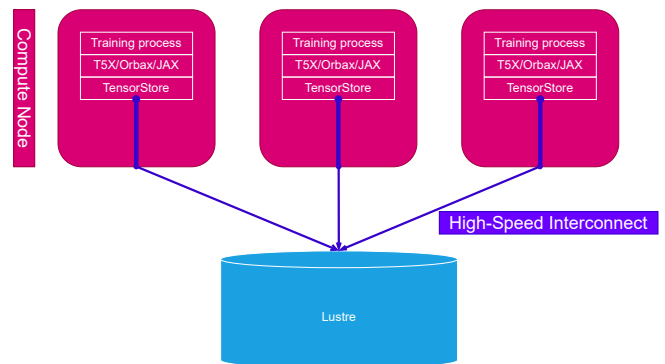


Fig. 1. Checkpoint process using Lustre filesystem.

```
'kvstore': { 'driver': 'file', 'path':
            '/path/to/dataset/', },
'driver': 'n5', 'metadata': { ... }, 'create':
            True, 'delete_existing': True, }).result()
dataset[80:82, 99:102].write([[1, 2, 3], [4, 5,
6]]).result()
```

Listing 1. The method of using TensorStore

### 3.2 CHFS

CHFS [6] is an ad hoc parallel file system that uses Optane Persistent Memory installed on compute nodes. CHFS constructs a parallel file system by executing a daemon process, chfsd, on each compute node.

## 4 DESIGN AND IMPLEMENTATION OF TENSORSTORE CHFS

Figure 1 depicts the checkpoint creation while training large language models using T5X. In HPC clusters, checkpoints were created on the parallel file system Lustre. TensorStore uses the “file” key-value store driver to create checkpoints on Lustre. The compute nodes and Lustre were connected using high-speed interconnects. However, the frequency of checkpoint creation is limited by the lack of write bandwidth of Lustre.

Figure 2 depicts the checkpoint creation process when TensorStore CHFS is used. TensorStore connects to the chfsd daemon via the CHFS client and sends the checkpoint data to chfsd. Then, chfsd writes the checkpoint data to the Intel Optane Persistent Memory installed on the compute node.

TensorStore natively supports multiple storage drivers, including Google Cloud Storage, local and network file systems, and in-memory storage systems. In this study, the CHFS was added to TensorStore as a key-value storage driver. The Listing 2 outlines the method to use TensorStore CHFS.

```
dataset = ts.open({
```

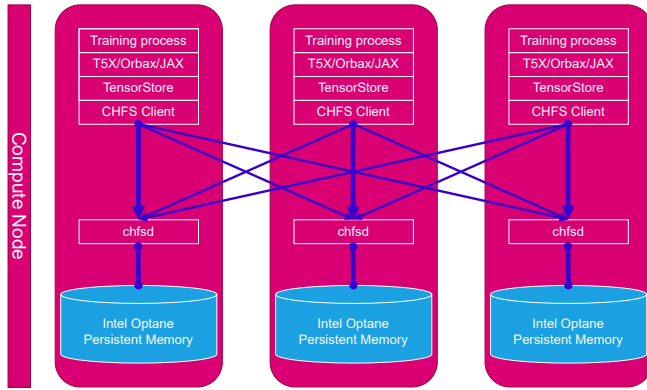


Fig. 2. Checkpoint process using TensorStore CHFS.

```
'driver': 'zarr', 'kvstore': { 'driver': 'chfs',
    'path': '/path/to/dataset/', },
}, dtype=ts.uint32, shape=[1000, 20000],
create=True).result()
```

Listing 2. The method to use TensorStore CHFS

TensorStore CHFS is based on the “file” key-value storage driver and uses the CHFS API instead of the Posix API. It majorly differs from the “file” key-value storage in that the “file” key-value storage has ACID guarantees; however, the TensorStore CHFS does not have ACID guarantees because it focuses on performance. ACID guarantees are not required for checkpoints creation.

## 5 EVALUATION

The bandwidth of checkpoint creation was measured with and without TensorStore CHFS using T5X. To measure the bandwidth, the parameter size of the T5 1.1 model was changed, and the model was divided into 8, 16, and 32 nodes. The corresponding checkpoint sizes were 115, 179, and 257GiB. The evaluation was performed using the Big Memory Supercomputer, Pegasus, at the University of Tsukuba. The Lustre file system was used for comparison with the TensorStore CHFS. Lustre is a 7.1 PB DDDN EXAScaler with an effective 40GB/s. Each compute node of the Pegasus is equipped with a 2TiB Intel Optane Persistent Memory 300 series and an Nvidia H100 Tensor Core GPU. The limit of the TensorStore parameter, `file_io_concurrency`, was set to 128.

The evaluation results are depicted in Figure 3. In the case of 32 nodes, the bandwidth of checkpoint creation with TensorStore CHFS was 78 GiB/s, which is 4.5 times that with Lustre.

## 6 CONCLUSION

In this study, we demonstrated that the checkpoint creation of large language models accelerated by adding the CHFS as a storage driver for TensorStore. Using our method, the bandwidth of checkpoint creation improved by 4.5 times during evaluation on 32 nodes.

## REFERENCES

- [1] BRADBURY, J., FROSTIG, R., HAWKINS, P., JOHNSON, M. J., LEARY, C., MACLAURIN, D., NECULA, G., PASZKE, A., VANDERPLAS, J., WANDERMAN-MILNE, S., AND ZHANG, Q. JAX: composable transformations of Python+NumPy programs, 2018.

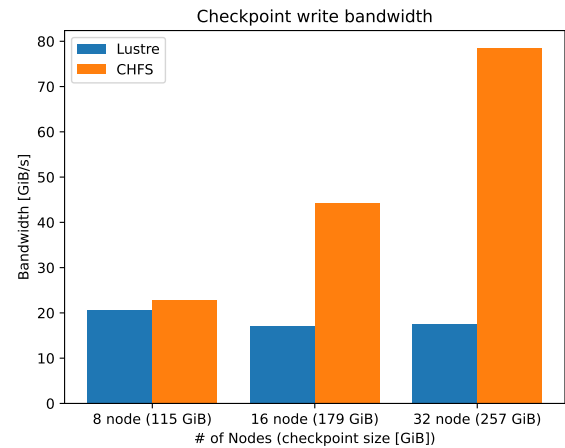


Fig. 3. Checkpoint write bandwidth.

- [2] EISENMAN, A., MATAM, K. K., INGRAM, S., MUDIGERE, D., KRISHNAMOORTHY, R., NAIR, K., SMELYANSKIY, M., AND ANNAVARAM, M. Check-N-Run: A checkpointing system for training deep learning recommendation models. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22) (2022)*, pp. 929–943.
- [3] GOOGLE. Tensorstore. <https://google.github.io/tensorstore/>, 2023. Accessed: July 31, 2023.
- [4] ORBAX. Orbax documentation. <https://orbax.readthedocs.io/en/latest/>, 2023. Accessed: July 31, 2023.
- [5] ROBERTS, A., CHUNG, H. W., LEVSKAYA, A., MISHRA, G., BRADBURY, J., ANDOR, D., NARANG, S., LESTER, B., GAFFNEY, C., MOHIUDDIN, A., ET AL. Scaling up models and data with t5x and seqio. *arXiv preprint arXiv:2203.17189 (2022)*.
- [6] TATEBE, O., OBATA, K., HIRAGA, K., AND OHTSUJI, H. Chfs: Parallel consistent hashing file system for node-local persistent memory. In *International Conference on High Performance Computing in Asia-Pacific Region (2022)*, pp. 115–124.
- [7] ZHAO, W. X., ZHOU, K., LI, J., TANG, T., WANG, X., HOU, Y., MIN, Y., ZHANG, B., ZHANG, J., DONG, Z., ET AL. A survey of large language models. *arXiv preprint arXiv:2303.18223 (2023)*.