EPJ Web of Conferences **214**, 05026 (2019) https://doi.org/10.1051/epjconf/201921405026
*CHEP 2018*

# The Event Buffer Management for MT-SNiPER

*Jiaheng* Zou[1,*], *Tao* Lin[1], *Weidong* Li[1], *Xingtao* Huang[2], *Ziyan* Deng[1], *Guofu* Cao[1], and *Zhengyun* You[3]

[1]Institute of High Energy Physics, Chinese Academy of Science, Beijing, China
[2]Shandong University, Jinan, China
[3]Sun Yat-sen University, Guangzhou, China

**Abstract.** SNiPER is a general purpose offline software framework for high energy physics experiment. It provides some features that are attractive to neutrino experiments, such as the event buffer. More than one events are available in the buffer according to a customizable time window, so that it is easy for users to apply events correlation analysis. We also implemented the MT-SNiPER to support multithreading computing based on Intel TBB. In MT-SNiPER, the event loop is split into pieces, and each piece is dispatched to a task. The global buffer, an extension and enhancement to the event buffer, is implemented for MT-SNiPER. The global buffer is available by all threads. It keeps all the events being processed in memory. When there is an available task, a subset of its events is dispatched to that task. There can be overlaps between the subsets in different tasks due to the time window. However, it is ensured that each event is processed only once. In the task side, the subsets of events are locally managed by a normal event buffer. So the global buffer can be transparent to most user algorithms. Within the global buffer, the multithreading computing of MT-SNiPER becomes more practicable.

## 1 Introduction

The SNiPER [1] framework is originally developed for the offline software of Jiangmen Underground Neutrino Observatory (JUNO) [2]. However, we have succeeded in stripping the dependencies and promoting it as a standalone project. SNiPER is very lightweight and easy to use. It's attractive to those who want a rapid integration of their software with a simple framework. Now SNiPER has been adopted by many more experiments, including Large High Altitude Air Shower Observatory (LHAASO) [3], China Spallation Neutron Source (CSNS) and Neutrinoless double beta decay experiment (nEXO) [4].

The data amount of our experiments will be very large in the future, which is a big challenge for us. It's necessary to consider any high performance computing techniques in the data processing software. Some parallel methods concentrate on accelerating the time consuming calculations in an application, such as NVIDIA CUDA and OpenMP. They are strongly coupled with the final software product. Some other methods provide high level

*e-mail: zoujh@ihep.ac.cn

abstraction of task parallelism, which can be used in a general purpose framework. GaudiHive [5] is a successful example in the high energy physics field.

Following the requirements and the trends in HEP software, MT-SNiPER has been developed for multithreading computing in 2017. The multiple task feature of SNiPER makes it easier to support multithreading without missing the simplicity. MT-SNiPER is implemented as a non-invasive wrapper of SNiPER kernel modules. So that it is almost transparent to most users, and the migration cost is minimized.

## 2 An overview of MT-SNiPER

MT-SNiPER is implemented based on Intel TBB. It is a combination of data parallelism and task parallelism. The conception of MT-SNiPER is straightforward. It mainly consists of 3 parts, including I/O tasks, global buffer and the Muster, as shown in Fig. 1.

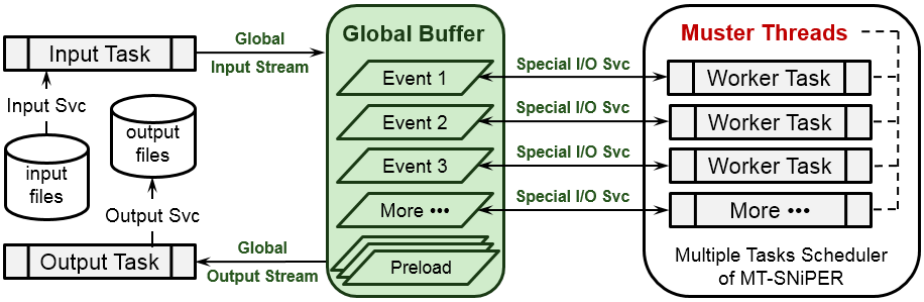


**Fig. 1.** The schematic diagram of MT-SNiPER.

The Muster module represents the data parallelism in MT-SNiPER. In high energy physics experiments, data consists of a serial of events. Generally events are independent to each other, and they can be handled concurrently. It is natural to apply the data parallelism model. At the beginning, Muster spawns a group of worker tasks and starts them in different threads. Then events are dispatched to workers and handled simultaneously. In each worker, it is a copy of the SNiPER kernel Task that performs like a serial job.

The I/O tasks represents the task parallelism. They are in charge of the events reading and writing on demand. In order to mitigate the I/O competing between threads, there is a global input task for each input stream, and a global output task for each output stream. The input task and output task are executed in separate threads besides the worker threads. Then the disk I/O synchronisation is much simplified in MT-SNiPER. We can directly reuse the serial version of I/O services in the global I/O tasks.

The global buffer decouples the disk I/O and worker tasks. On the worker side, event references are transferred via customized I/O services between global buffer and workers. Since the reference management in memory is much faster than disk I/O, the locks for data synchronisation can be very thin. On the I/O task side, semaphores are used to trigger the reading or writing procedure. Events in global buffer are kept in its reading order. So the result in global buffer can be written out directly without re-sorting.

Event level parallel computing is achieved in MT-SNiPER. The SNiPER Tasks are simply mapped to Intel TBB tasks. This is a splendid practice of the SNiPER multiple task feature in multithreading context.

## 3 The event buffer of MT-SNiPER

The event management in memory is an important part of MT-SNiPER. It is the key of synchronisation between threads. And it became more complicated when we introduced support for events correlations for neutrino experiments.

### 3.1 Global buffer

The global buffer is indeed a FIFO queue. Events are read into the buffer serially, and wrote out in the same order after processing. It interacts with all the I/O tasks and workers. Thread safety should be managed carefully. It is implemented as a ring for simplicity and better performance, as shown in Fig. 2.

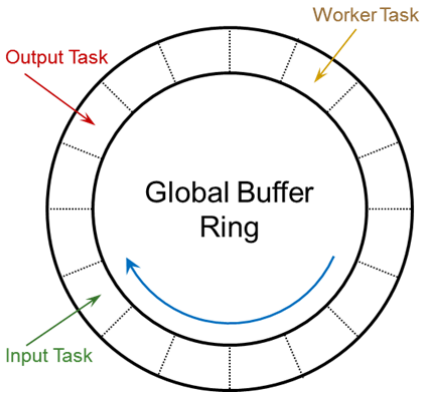


**Fig. 2.** The global buffer ring in MT-SNiPER.

There are 3 pointers associated with the global buffer ring. The pointers are respectively accessed by the input task, the workers and the output task.

The 1st pointer indicates the end of the event queue. When an event is read, it is filled in the position of this pointer. Then the pointer moves to the next position, and a waiting worker is notified. The input task will be paused when the ring buffer is full.

The 2nd pointer indicates the next event to be dispatched to a worker. Workers will be paused when this pointer catch up with the 1st one in the ring. The output task is notified when an event is processed. We have to prevent concurrent accessing of the pointer by different workers.

The 3rd pointer indicates the beginning of the event queue. When the first event is processed, it is sent to the output task for writing and removed from the queue. The input task is notified when the number of empty slots in the buffer is greater than a safety value. This procedure is paused until the first event processing hasn't been finished.

The capacity and safety value of the ring buffer can be configured at the beginning of a job. For different type of jobs, it should be optimized according to the memory usage and threads competing.

### 3.2 Local buffer in a worker

Customized I/O services are implemented for MT-SNiPER workers. The input service gets event references from the global buffer instead of reading files. The output service sets a state to the processed events instead of writing them out. Each worker is limited to the scope of a SNiPER Task. It is the same as a serial SNiPER job in most cases.

However, it is a little difference when there are events correlations. In this case, a group of continuous events are dispatched to a worker each time. There can be overlaps between event groups. Additional event state is used to ensure each event is processed only once.

## 4 Performance measurement

We took a preliminary measurement to estimate the performance of the global buffer in MT-SNiPER. Following are the execution environment and testing case,

- A node with 32 CPU cores
- An algorithm that takes an average of 0.01 second per event

The speedup ratio is very close to the ideal value when we use no more than 20 worker threads, as shown in Fig. 3.

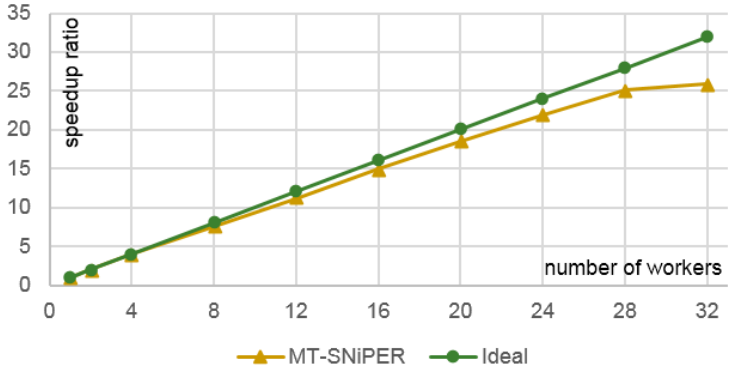


**Fig. 3.** A preliminary result for the speedup radio of MT-SNiPER

## 5 Conclusions

SNiPER is a simple and lightweight software framework which has been used by several experiments. It is enhanced by MT-SNiPER for multithreading computing. In MT-SNiPER, a global buffer is implemented to decouple the disk I/O and workers. Event references are mapped into workers' local buffer and handled concurrently. There is only a few thin locks in the optimized global buffer. A preliminary testing shows that the overhead is very small. The hierarchic buffer architecture is an effective solution in MT-SNiPER.

This work is supported by Joint Large-Scale Scientific Facility Funds of the NSFC and CAS (Grant No. U1532258), the Youth Innovation Promotion Association of Chinese Academy of Sciences (Grant No. 2017021), the National Natural Science Foundation of China (Grant No. 11605221) and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA10010900).

## References

1. Zou J H, Huang X T, Li W D, Lin T, Li T, Zhang K, Deng Z Y and Cao G F 2015 *J. Phys.: Conf. Ser.* **664** 072053. See also "SNiPER" [software], version 1.0, Available from https://github.com/SNiPER-Framework/sniper/releases/tag/v1.0 [accessed 2018-03-20]
2. An F *et al.* (JUNO) *J. Phys.* **G43** 030401 (2016)
3. Cao Zhen *et al.* A future project at tibet: the large high altitude air shower observatory (LHAASO) *Chinese Phys. C* **34** 249 (2010)
4. The nEXO Collaboration, nEXO Pre-Conceptual Design Report, arXiv:1805.11142v2 [physics.ins-det] (2018)
5. M. Clemencic *et al. J. Phys.: Conf. Ser.* **513** 022013 (2014)