Radiation Detection Technology and Methods (2024) 8:1704–1711
https://doi.org/10.1007/s41605-024-00477-6

ORIGINAL PAPER

Event index-based analysis in the JUNO experiment

Yixiang Yang[1,2] · Weidong Li[1,2] · Tao Lin[1] · Jiaheng Zou[1] · Simon Charles Blyth[1] · Xingtao Huang[3] · Teng Li[3]

Received: 19 April 2024 / Revised: 19 April 2024 / Accepted: 10 May 2024 / Published online: 20 June 2024
© Institute of High Energy Physics, Chinese Academy of Sciences 2024

Abstract

Purpose The Jiangmen Underground Neutrino Observatory (JUNO) is a large-scale neutrino experiment designed mainly for determining the neutrino mass hierarchy and accurately measuring the parameters of neutrino oscillation by detecting reactor antineutrinos. It is estimated that the detector will record approximately 5.2 TB of raw data every day, which only contain around 60 neutrino events. The time correlation analysis needs to select physics events from a very large data set.

Methods To address this challenge, an event index-based analysis software framework is designed and its major developments include: 1) A new analysis workflow is implemented including pre-selection and event selection, and the former works as a data filter of the latter. 2) A new data format is defined to support reading indexed events efficiently for event pre-selection. In order to further improve the performance of data analysis, other developments are also completed, i.e. adding the support of multithreading based on TBB and moving computing-intensive components from data I/O service to the computation algorithm.

Results and conclusion The performance tests are completed on the simulated data which are close to real experimental data. The results show that reading data are 5 times faster from an IAD file than from a tree in the ROOT file. The multithreaded analysis exhibits high scalability. Performance studies also show that, with the event index method and multithreading, the efficiency of a typical analysis of JUNO can be improved by two orders of magnitude.

Keywords JUNO · Indexed data · Data analysis · Multithreading

Introduction

JUNO detector

The Jiangmen Underground Neutrino Observatory (JUNO) [1] is a large-scale neutrino experiment targeting at an abundant physics research program. The experiment mainly focuses on determining the neutrino mass hierarchy, measuring neutrino oscillation parameters with unprecedented precision, probing neutrinos from supernovae and the Sun, and potentially contributing to the search for dark matter [2]. JUNO is currently constructed in Jiangmen, southern China. Being approximately 53 km away from the Yangjiang and Taishan nuclear power plants, the facility is strategically located at an underground site with an overburden of roughly 650 m to shield against cosmic muons. As shown in Fig. 1, the JUNO detector comprises three sub-detectors that are the central detector (CD), the water pool (WP), and the top tracker (TT), respectively. The CD is a liquid scintillator (LS) detector, containing 20,000 tons of LS in a 35.4-m-diameter

✉ Weidong Li
liwd@ihep.ac.cn

Yixiang Yang
yangyixiang@ihep.ac.cn

Tao Lin
lintao@ihep.ac.cn

Jiaheng Zou
zoujh@ihep.ac.cn

Simon Charles Blyth
blyth@ihep.ac.cn

Xingtao Huang
huangxt@sdu.edu.cn

Teng Li
tengli@sdu.edu.cn

[1] Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China

[2] University of Chinese Academy of Sciences, Beijing 100049, China

[3] Shandong University, Qingdao 266237, China

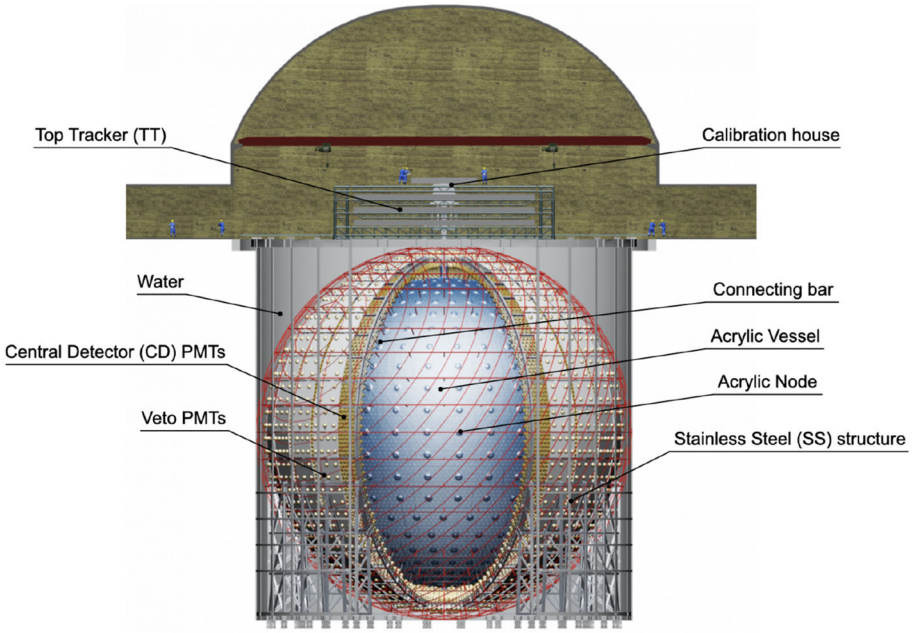

Fig. 1 Schematic view of the JUNO detector [4]

spherical acrylic vessel which is supported by the stainless steel (SS) structure with an inner diameter of 40.1 m. The CD is surrounded by a cylindrical water pool containing 35 kilotons of ultra-pure water, which acts as a buffer against radioactivity from the rocks. The connecting bar connects the SS structure and the acrylic node on the vessel. The photons emitted from the LS are measured by an array of photomultiplier tubes (PMTs) installed on the inner side of the SS structure, including 17,612 20-inch PMTs and 25,600 3-inch PMTs. Additionally, the WP is equipped with 2400 20-inch Veto PMTs, serving as a Cherenkov detector to reject cosmic muons. In order to precisely measure the direction and further veto cosmic muons, the TT, a plastic scintillator array, is installed on top of the WP. The TT consists of 62 walls, each with a sensitive area of $6.7 \times 7.7\,\mathrm{m}^2$, and covers about 60% of the WP. There is also a Calibration House to accommodate the well-designed calibration system. With a novel dual calorimetry technique, calibration can achieve at least 1% energy linearity and 3% energy resolution [3].

Data processing

During data taking, data from the JUNO detector will be buffered in the detector's front-end electronics and only those that met the trigger criteria will be read out by the Data Acquisition (DAQ) at a rate of 40 GB/s [5]. In the DAQ, the triggered data from each sub-detector will be assembled into events so that online event classification and software trigger can be performed to further reduce data volume by 500 times. The recorded data are in byte stream format which is so-called Raw data. All the Raw data will be transferred to

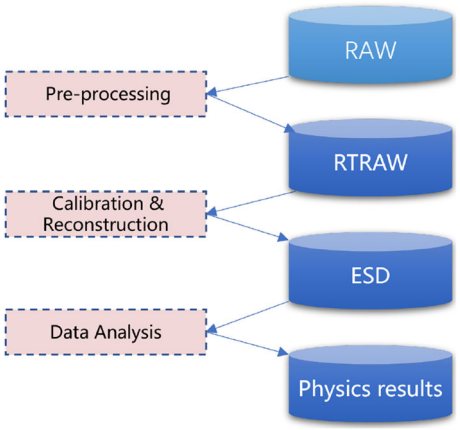

Fig. 2 Offline data processing flow

the IHEP data center via a dedicated network with a bandwidth of a few Gbps.

The Raw data is processed and reconstructed in the computing cluster at IHEP, and the generated data will be used as the input of physics analysis. As illustrated in Fig. 2, the offline data processing contains three consecutive steps that are data pre-processing, calibration, and reconstruction. In the data pre-processing, events will be sorted according to their timestamps and converted to RTRaw data, which is of ROOT format [6]. The contents of RTRaw data are identical to those of Raw data which mainly contain time and charge information of the detection modules like PMTs in CD and WP. In the calibration, the detector response will be calibrated at different levels from electronics channels to overall detector performance such as energy non-linearity and spatial non-uniformity of the detector response. In the

reconstruction, physics quantities will be calculated, for example, reconstructed energy and position of an incident particle. The results of reconstruction will be saved in ROOT files which are referred as Event Summary Data (ESD) data files.

Physics analysis

JUNO detects electron antineutrinos via the IBD interaction, $\overline{\nu}_e + p \rightarrow e^+ + n$. The e^+ quickly deposits its energy and annihilates into two 0.511-MeV photons, which provide a prompt signal. After approximately 200 μs (on average) of scattering in the detector, the neutron is typically captured by a proton and releases a 2.2 MeV gamma forming the delayed signal.

One of the major tasks of event analysis is to select electron antineutrino events and determine the mass ordering according to its energy spectrum [2]. A physics event of antineutrino consists of a prompt signal and a delayed signal which are spread in two separate readout events. The event selection is done by using a so-called time correlation analysis method as shown in Fig. 3. As mentioned above, all the readout events are sorted by time before the data analysis starts. In the time correlation analysis, event selection will be performed to the readout events within a user-defined time window. The event in the middle of the window is called the current event. If the current event is identified as the delayed signal, the search for the prompt signal will take place among the readout events ahead of the current event. Otherwise, the search for a delayed signal will take place in the opposite direction. By sliding the window, event selection can go through all the events.

In order to support data processing and analysis, the SNiPER (Software for Non-collider PhysIcs expERiment) software framework [7] has been developed. With the SNiPER, all functional modules such as algorithms, tools, and services can be implemented in C++ and the data processing flow is controlled with Python configuration scripts. Event data model [8] is developed to describe the event data produced in different processing stages. An event buffer with the time window is implemented to manage event data objects created by the associated data input service. The user-defined time window is supported to facilitate the time correlation analysis on multiple readout events. The implementation of concurrency is based on Threading Building Blocks (TBB) [9], and algorithms can be executed on multiple CPU threads in parallel.

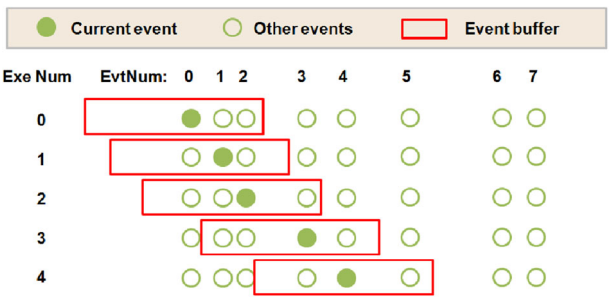

Fig. 3 Time correlation analysis

Analysis requirements

The main expected background sources in the JUNO are cosmic muons, fast neutrons, cosmogenic isotopes induced by muon spallation in the liquid scintillator, and natural radioactivity. Natural radioactivity mainly comes from the materials of the detector and materials in the environmental hall. Although very strict control has already been established and implemented in the construction and installation of the detector, the radioactivity background still dominates the event rate due to the huge amount of material inside the detector. When only using single event selection criteria, it is difficult to distinguish the IBD signals from the background, since they are quite similar. The correlation between prompt and delayed signals is the prominent feature of IBD event. So the correlation analysis is needed to drastically reduce background from radioactivity.

According to a previous study at JUNO [10], the event rate can reach 88 Hz even if the energy threshold for the trigger is set to a value that corresponds to 600 fired PMTs and the fiducial volume is considered to have the radius of 17.2 m. In fact, for future data taking, the trigger threshold may be looser, and the event rate may be higher. Since the event rate for physics channels is very low, i.e., about 60 IBD events/day, the physics events which are interesting to the physicists are very sparse.

Considering the characteristics of the experimental data, the analysis software needs to have the following functionalities: (1) Efficiently searching for the sparse physics signals in large volumes of data. (2) Straightforwardly performing time correlation analysis to select neutrino events by rejecting background events that have similarities to the physics signals. (3) Easily applying a second event reconstruction with an optimized algorithm to improve event selection performance. In the current JUNO software release, the time correlation analysis has already been supported. Owing to the SNiPER, the user only needs to implement the event selection algorithm for the physics channel being studied. A lazy loading mechanism [11] using lightweight headers and data objects has been implemented in the SNiPER to only load objects required by the analysis. Since the ESD also contains CalibHits data, a refined event reconstruction can also be applied to the selected events during the analysis. As to the requirements of analysis, the first item, efficient search

for physical signals, remains an urgent issue that needs to be solved.

IAD-based analysis

Workflow design

The Indexed Analysis Data (IAD) is an event data type defined to facilitate quick search for neutrino events in a very large data set. An IAD event comprises two components: (1) the necessary physics information required for rapid event pre-selection and (2) an index pointing to its associated ESD event. Both IAD and ESD will be generated during data reconstruction. Compared with an ESD event, the size of an IAD event is much smaller because it only contains some key variables of an event such as event timestamp and reconstructed vertex and energy. Each IAD event has an index to its corresponding ESD event. The IAD event and the ESD event are just two different representations of the same event.

The IAD-based analysis follows a two-step scheme, as depicted in Fig. 4. As the first step, a block of IAD events is loaded, and a rapid event pre-selection is performed. If the pre-selection criteria are not satisfied, the analysis will move on to the next IAD event block. Otherwise, its corresponding ESD events sharing the time window of the selected IAD event will be loaded into memory using an index which is an address pointing to the ESD event. Then, the second step starts, in which analysis will be applied to the ESD events in the memory.

In the event pre-selection step, time correlation analysis will be done with minimum information of the event. The pre-selection criteria are used to identify and select the events that might belong to a physics event. For example, the selected event could be a candidate of a prompt signal or a delayed signal of a neutrino physics event. The event pre-selection is designed to reduce access to ESD, so the efficiency for pre-selection should be very close to 100% which means no physics signals will be lost.

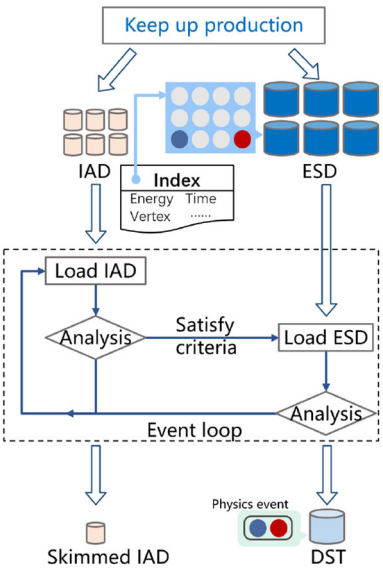

Fig. 4 Workflow of IAD-based analysis method

In the event selection step, time correlation analysis is applied using the complete information of events stored in the ESD. Since the ESD also contains CalibHits, it is feasible to perform a refined reconstruction according to event type at this step. By using results from refined reconstruction, analysis can obtain better performance at the cost of more CPU time consumed in the refined reconstruction.

The event pre-selection and event selection have different roles in the data analysis. By applying event pre-selection, access to ESD can be reduced greatly, and the overall performance of the analysis will be improved. The event pre-selection and event selection can be executed either in a single job or in two separate jobs. From the software point of view, merging two tasks into a single workflow will be a good solution to support both use cases.

Software implementation

One of the goals is to build a solution for users by being compatible with their existing analysis code, which means no changes need to be done to the user's code when they switch to the new analysis environment.

In the SNiPER framework, the existing event selection performed on ESD is implemented as an algorithm and it retrieves event data objects, defined by the event model, from an ESD buffer. When the event selection algorithm requests event data objects, they are created by the data input service by reading the persistent event data stored in a file, converting it to transient data objects, and registering them in the buffer.

The implementation of IAD-based analysis is based on SNiPER, and the workflow is shown in Fig. 5. In order to fit the scheme of existing implementation, the event pre-selection will be added and is also implemented as an algorithm. It works as a data filter of the data input service of event selection. After adding event pre-selection, when the event selection requests event data from the ESD buffer, the input service will not be invoked until the event pre-selection provides it the address of the event data (the Index). In order to achieve this, IAD events will be read into the Global buffer and analyzed by the event pre-selection. If an event meets the selection criteria, the address of its associated ESD event will be returned to the input service of event selection so that the event selection can resume.

For a user who wants to do IAD-based analysis, he only needs to provide his existing analysis program plus a new "IAD Alg" to describe how to select IAD events. The IAD-based analysis framework will handle the control flow of the analysis and do all the rest.

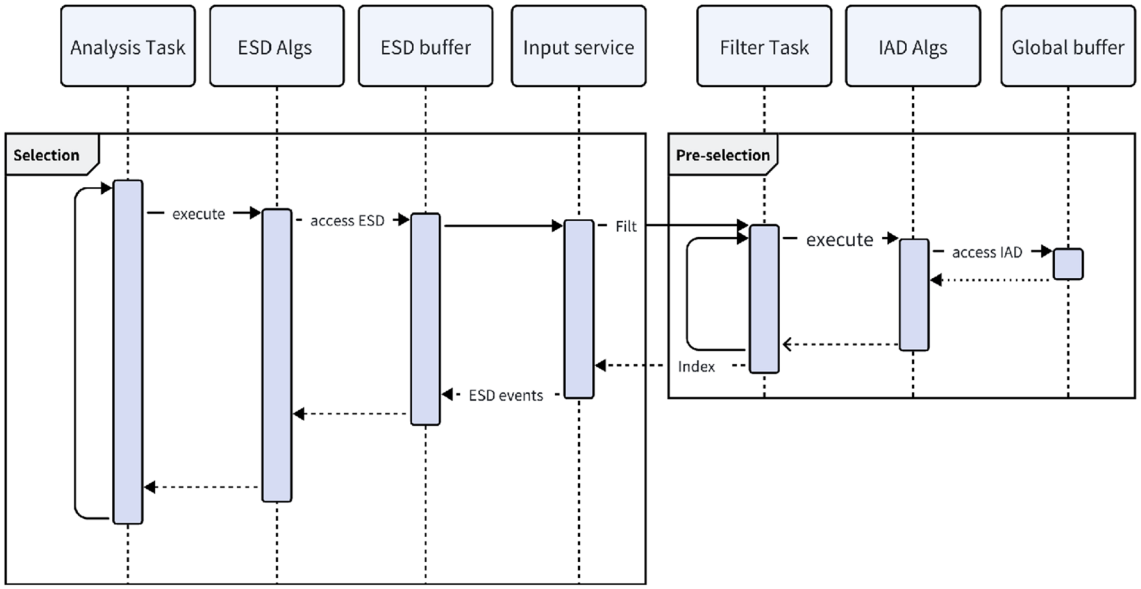

Fig. 5 Implementation of IAD-based analysis with a analysis filter structure. It is a software sequence diagram that shows the workflow. The solid and dashed arrows show the calling relationships between modules at runtime. The software includes the selection part and pre-selection part. The pre-selection part works as a filter and is driven by the selection part

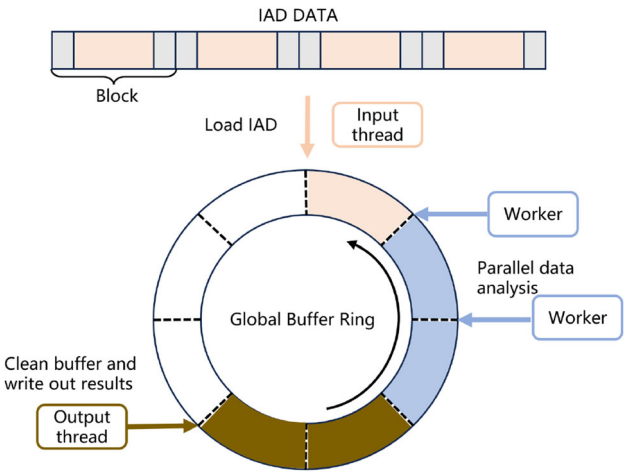

Fig. 6 Multithreaded IAD-based analysis. The ring buffer is used to synchronise the threads. All the threads go around the ring counter-clockwise. The colored arrow indicates which cell the thread is perching on. Each cell of the global buffer can hold a block. For the colors of the cell in the ring, white represents the cell does not hold a block. Pink represents the block waits for processing. Blue represents the block is being processed by the worker. Brown represents the block is ready to be cleaned by the output thread

Concurrency

Multithreaded analysis

In a multithreaded application, all the threads in the process share the same address space; no extra overhead will be introduced in exchanging data between different threads. Compared with multi-processing, multithreading has a higher efficiency in program execution. So the implementation of analysis concurrency is based on multithreading which is built on TBB. In the implementation, both event pre-selection and event selection are combined into a processing sequence which is performed by one thread. Since a thread can run on a single CPU core, event pre-selection and event selection can be executed concurrently within a multi-core job.

Data analysis application needs to process large volumes of data typically terabytes or petabytes in size. For a multithreaded analysis, it helps to improve the analysis performance to dispatch event blocks rather than a single event to a thread. So the event block consisting of many events is considered as the basic data unit for processing. As shown in Fig. 6, the IAD block, event block of the IAD type, is designed to have a sandwich structure, where the middle part (in pink color) is events in the central region of the block, and the left and right parts (in gray color) represent events in the edge of the block which is overlapped with its neighboring block. By choosing the proper size of the edge, time correlation analysis can be applied to the events in the central region.

For multithreading, the disadvantage is race condition which will occur when more than one thread is required to alter the shared data at the same time. To prevent this, thread synchronization is needed. As shown in Fig. 6, the "Global buffer" is used to synchronize the multiple threads. The input thread is responsible for reading IAD blocks, each containing a hundred of thousands of events, from files and then filling the global buffer, until the global buffer is full. Every time the input thread fill the buffer, it notifies the Worker to process. The Worker thread (Worker) retrieves these blocks from the

global buffer and performs the two-step IAD-based analysis. Once the Worker finishes one block, it will notify the output thread to clean the global buffer and write out the results. All the threads move around the Global buffer ring in the same direction, and the Global buffer also ensures the results are written out in order.

Improving data throughput

Data analysis normally belongs to the category of data-intensive applications that have large volumes of data as the input. In JUNO, time correlation analysis devotes its processing time not only to reading data but also to processing data, e.g., second round of event reconstruction taking place during data analysis. On a multi-core processor, data processing in the analysis can be accelerated just simply by increasing the number of threads. So in the end, how to increase the input capability of data analysis becomes a critical path item on improving the overall performance.

A new data format called IAD format is defined to support reading data from a file in an efficient way. In this format, IAD is mainly a collection of event blocks, each of which contains a hundred of thousands of events. Both the event block and the event itself have a header data structure. The header contains the metadata, such as the index. The physics information is stored in the data part of the event. Users are allowed to customize the physics information by defining their own IAD event data model, which can be derived from TObject or simply be of a plain old data (POD) type. In the case of TObject, a complex structure can be defined using object-oriented features. In the case of POD, the data structure is only represented as passive collections of field values (instance variables). In both cases, data compression and decompression are supported.

According to a study on the optimization for ROOT IO [12], separation of data I/O from computing and making computing parallel can improve data loading throughput. When reading data from a file, data decompression and creation of IAD objects are quite computing intense compared with the process of data loading. So the execution of the computing-intense part is moved to the Worker thread to achieve a decoupled data loading. In the optimized implementation, the input thread is only responsible for copying event blocks containing compressed event data from the file to the Global buffer.

Performance

Since JUNO has not taken data yet, the data used in the performance test are simulated data that are close to real experimental data. Firstly, the Geant4-based detector simulation is used to generate IBD events (JUNO main signals), cosmic muon events, and natural radioactive background events, separately. Then, simulated events are mixed at their theoretical rates, and then, the mixed events are passed to electronics simulation (including trigger) and event reconstruction.

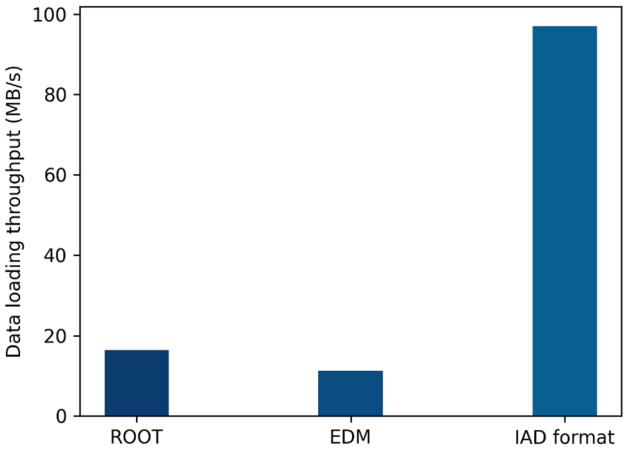

Fig. 7 Data loading performance of IAD format compared to that of ROOT and ESD formats. The data loading also includes the deserialization process where data are decompressed; event objects are constructed. During the test, we record the time duration from beginning of the data loading to the time point when the all the event objects are ready in the memory. The throughput is calculated from this time duration

After the trigger, the total event rate is about 100 Hz and very few IBD events will be generated. So, the IBD rate is enlarged and set to 5 Hz for the purpose of performance evaluation. The IAD sample is generated by reproducing the ESD sample and writing out the needed reconstruction information as well as the event index.

All tests were carried out on a dedicated workstation. The detailed configurations are as follows:

- Intel(R) Xeon(R) Silver CPU, 24 cores, with Hyper-Threading and Turbo-Boost off
- 128 GiB memory
- CentOS 7.9 with JUNO official offline software environment
- HGST hard disk drive, 7200 RPM

To evaluate the performance of loading data, the IAD samples in different formats are generated such as ROOT, ESD, and IAD format. For the ROOT format, the IAD events are stored in a tree with the split level as zero. For the ESD format, both navigators, data objects used to manage readout events in ESD, and events are stored in different trees, respectively. All the data samples are compressed with the same compression algorithm.

As shown in Fig. 7, the speed of loading data in IAD format is approximately 5 times faster than that in ROOT format. This is owing to the characteristics of the IAD format

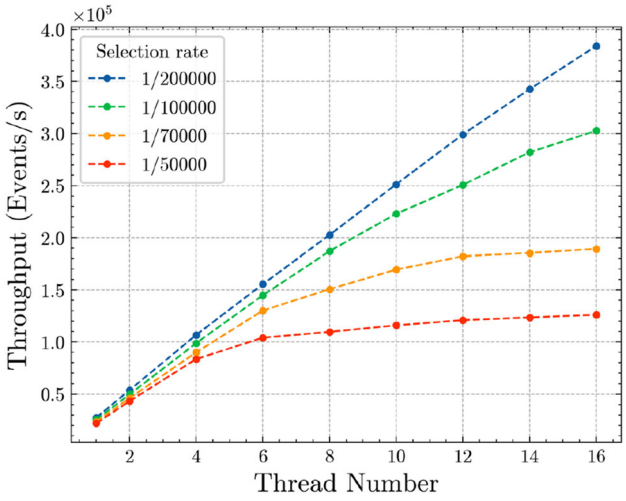

Fig. 8 Scalability of multithreaded IAD analysis. The dummy algorithm is used, whose performance mimics that of a real IBD selection algorithm. It takes about 0.03 ms to process a event. The selection rate refers to the rate at which the IAD events pass the selection. For each selected IAD event, all the ESD events in the time window are loaded. Each time window contains about 10 ESD events

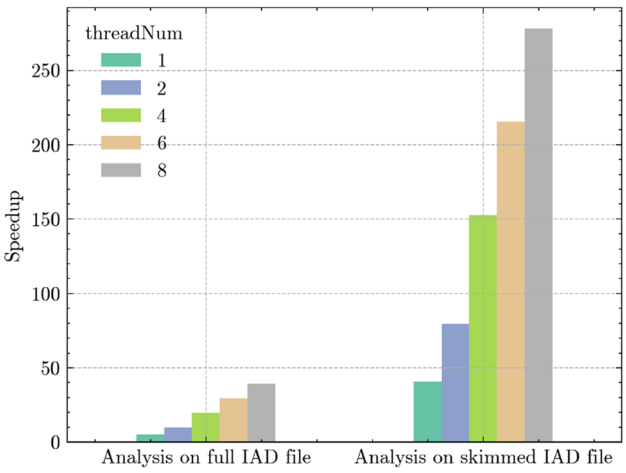

Fig. 9 Framework performance on real IBD selection. A set of preliminary antineutrino selection cuts including muon veto are used in the algorithm. The algorithm takes about 0.03 ms to process one event

which allows sequentially reading large block data, as well as owning a much simpler deserialization process compared to the ROOT format. The data loading speed for the ESD format is roughly 70% of that for the ROOT format. That is because the ESD format has a more complex structure. The process is: the navigator of the event is first read into memory, and then, the event is accessed through the SmartRef provided by the navigator. So reading ESD events costs an additional overhead time.

To evaluate the scalability of the multithreaded IAD analysis framework, we run dummy analysis algorithms in the framework with different numbers of threads and different selection rates. The data throughput versus the number of threads is shown in Fig. 8. It can be seen that the data throughput is also related to the selection ratio representing the frequency by which the ESD is accessed. If the selection rate is lower than 1/200,000, the performance scales linearly with the number of threads. Since the signal ratio is much less than 1/200,000 in the real data, the multithreaded IAD analysis can achieve good scalability in actual use.

We also perform performance tests with a real IBD selection algorithm. The performance comparison is shown in Fig. 9. It can be found that the speedup of analysis is increased with the number of threads. With a single thread being used, the IAD-based analysis is about 4 times faster than the analysis without pre-selection. By applying data skimming on IAD blocks, the IAD sample is reduced to 12% of its original size. By checking Monte Carlo truth, it is found that the skimmed IAD sample contains the same IBD events as the original IAD sample does. When the IAD-based analysis is applied to the skimmed IAD sample, a speedup of 40 times can be achieved. And when using multi-threading, the speedup can reach up to two orders of magnitude.

Conclusion

In this paper, we present an event index-based analysis software framework to facilitate time correlation analysis for the JUNO experiment. The whole analysis is divided into two consecutive steps: event pre-selection and event selection. At the pre-selection step, the number of events is reduced by analyzing IAD events and applying pre-selection cuts on IAD events. If the pre-selection criteria are satisfied, the analysis moves to event selection by loading the ESD event using the event index bound in the IAD event. The event pre-selection is implemented as a data filter which can be smoothly integrated into the current analysis software.

In order to improve the performance of the data analysis, further development is completed by adding the support of concurrency and improving data throughput. Multithreaded analysis is developed by combining event pre-selection and event selection into a sequence and several analysis sequences are executed on threads in parallel. A new data format called IAD format, consisting of a collection of event blocks each of which contains a hundred of thousands of events, is defined to support reading data from a file in an efficient way. Furthermore, execution of computing-intense part in the data thread is moved to the worker thread to separate computing from loading data.

In the application of the software framework, the IAD will be produced with the ESD in the data reconstruction process. The IAD events and ESD events are in one-to-one correspondence. By performing event pre-selection on IAD,

the access to ESD, which is heavy and time-consuming, is reduced to a negligible level.

With the simulated data, performance tests have been completed. The results show that reading data is 5 times faster from an IAD file than from a tree in the ROOT file, which is a big improvement in data I/O. The multithreaded analysis exhibits high scalability, and no significant performance loss caused by thread synchronization is found. The IBD event selection using the IAD-based framework is 4 times faster than the event selection with no IAD pre-selection being applied. A subset of IAD, whose volume is 12% of that of the original one, can be obtained by performing IAD skimming. By using a skimmed IAD dataset, approximately 40 times speedup can be obtained.

Acknowledgements This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA10010900, National Natural Science Foundation of China (NSFC) under Grant No. 12375195, No. 12025502, No. 12341504, and Youth Innovation Promotion Association under Grant No. 2022011.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. T. Adam, F. An, G. An, Q. An, N. Anfimov, V. Antonelli, G. Baccolo, M. Baldoncini, E. Baussan, M. Bellato, et al.: Juno conceptual design report. arXiv preprint arXiv:1508.07166 (2015)
2. F. An, G. An, Q. An, V. Antonelli, E. Baussan, J. Beacom, L. Bezrukov, S. Blyth, R. Brugnera, M.B. Avanzini et al., Neutrino physics with juno. J. Phys. G: Nucl. Part. Phys. **43**(3), 030401 (2016)
3. A. Abusleme, T. Adam, S. Ahmad, R. Ahmed, S. Aiello, M. Akram, F. An, G. An, Q. An, G. Andronico et al., Calibration strategy of the juno experiment. J. High Energy Phys. **2021**(3), 1–33 (2021)
4. R. Marina, Jiangmen underground neutrino observatory (juno): on the way to physics data (2022)
5. A. Abusleme, T. Adam, S. Ahmad, R. Ahmed, S. Aiello, M. Akram, F. An, G. An, Q. An, G. Andronico, et al.: Juno physics and detector. arXiv preprint arXiv:2104.02565 (2021)
6. I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun, N. Buncic, P. Canal, D. Casadei, O. Couet, V. Fine et al., Root-a c++ framework for petabyte data storage, statistical analysis and visualization. Comput. Phys. Commun. **180**(12), 2499–2512 (2009)
7. J. Zou, X. Huang, W. Li, T. Lin, T. Li, K. Zhang, Z. Deng, G. Cao, Sniper: an offline software framework for non-collider physics experiments. J. Phys.: Conf. Series **664**, 072053 (2015)
8. T. Li, X. Xia, X.-T. Huang, J.-H. Zou, W.-D. Li, T. Lin, K. Zhang, Z.-Y. Deng, Design and development of juno event data model. Chin. Phys. C **41**(6), 066201 (2017)
9. J. Reinders, Intel threading building blocks - outfitting C++ for multi-core processor parallelism (2007)
10. X. Fang, Y. Zhang, G. Gong, G. Cao, T. Lin, C. Yang, W. Li, Capability of detecting low energy events in juno central detector. J. Instrum. **15**(03), 03020 (2020)
11. T. Li, X. Huang, A new type of smart pointer for data object reference both in memory and in root files. J. Phys.: Conf. Series **762**, 012001 (2016)
12. B. Bockelman, Z. Zhang, J. Pivarski, Optimizing root io for analysis. J. Phys.: Conf. Series **1085**, 032012 (2018)

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.