Offline Data Processing Software for the Super Tau Charm Facility

Teng Li[1,*], *Wenhao* Huang[1], *Xingtao* Huang[1,**], *Xiaocong* Ai[2,***], *He* Li[3], and *Dong* Liu[3]

[1]Key Laboratory of Particle Physics and Particle Irradiation (MOE), Institute of Frontier and Interdisciplinary Science, Shandong University, Qingdao, Shandong, 266327, China
[2]School of Physics and Microelectronics, Zhengzhou University, Zhengzhou, Henan, 450001, China
[3]University of Science and Technology of China, Hefei, 230026, China

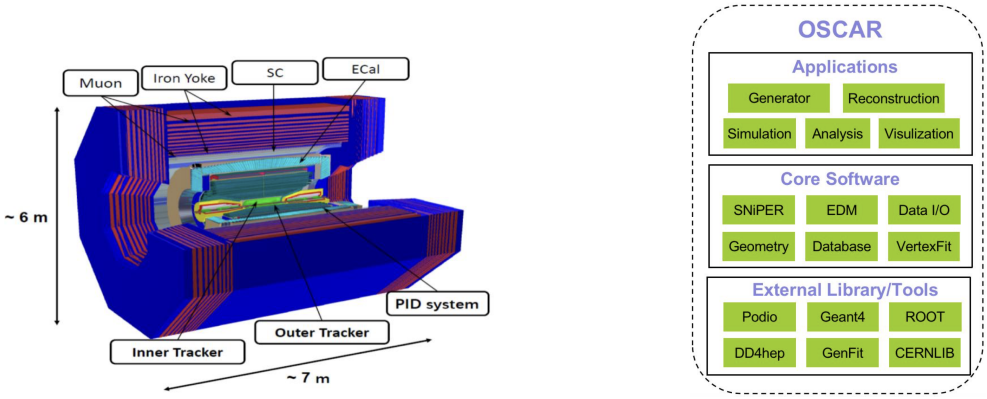
Abstract. The Super Tau Charm Facility (STCF) proposed in China is a new-generation electron–positron collider with center-of-mass energies covering 2-7 GeV and a peak luminosity of 0.5×10^{35} cm^{-2} s^{-1}. The offline software of STCF (OSCAR) is developed to support the offline data processing, including detector simulation, reconstruction, calibration as well as physics analysis. To meet STCF's specific requirements, OSCAR is designed and developed based on the SNiPER framework, a lightweight common software for HEP experiments. Besides the commonly used software such as Geant4 and ROOT, several state-of-the-art software packages and tools in the HEP community are incorporated as well, such as the Detector Description Toolkit (DD4hep), the plain-old-data I/O (podio) and Intel Thread Building Blocks (TBB) etc. This paper will present the overall design as well as some implementation details of OSCAR, including the event data management, paralleled data processing based on SNiPER and TBB as well as the geometry management system based on DD4hep. Currently, OSCAR is fully functioning to facilitate the conceptual design of the STCF detector and the study of its physics potential.

1 Introduction

The Super Tau-Charm Facility (STCF) [1, 2] is an electron-positron collider proposed with the center-of-mass energy ranging from 2 to 7 GeV, the transition region between perturbative and non-perturbative quantum chromodynamics. The peak luminosity of STCF is designed to be at 0.5×10^{35} cm^{-2} s^{-1}, which is almost two orders of magnitude higher than the present Tau-Charm factory in China. STCF is designed to study tau-charm physics, hadron physics as well as searching new physics beyond the Standard Model. The offline software of STCF (OSCAR) [3] is developed to facilitate the offline data processing tasks, including the production of Monte-Carlo simulation data, calibration and reconstruction of collected data, as well as helping physicists to conduct physics analysis, etc. To overcome the great challenges posed by the high luminosity of STCF, the OSCAR system is carefully designed and implemented to provide excellent performance for offline data processing tasks.

*e-mail: tengli@sdu.edu.cn
**e-mail: huangxt@sdu.edu.cn
***e-mail: xiaocongai@zzu.edu.cn

The overall architecture of OSCAR is shown in Figure 1. The OSCAR system is composed of three layers: the bottom layer includes some commonly used libraries and interfaces to the frequently used software and tools in the HEP community, such as ROOT [4], Geant4 [5], as well as some of the Common Turnkey Software Stack (Key4hep) [6], such as podio [7] and DD4hep [8]. The core software layer includes the underlying framework, and other common components used in offline data processing, such as the event data model (EDM), event data and detector data management systems, providing key and common functionalities for the offline software. The application layer includes the STCF-specific applications, such as the implementation of physics generators, detector simulation, digitization, reconstruction etc.



(a) (b)

Figure 1: (a) Schematic layout of the STCF detector. (b) Overview of the OSCAR system.

This paper aims to introduce the design and implementation of the OSCAR system briefly. Section 2 gives brief introduction of the underlying framework. In section 3, the event data and detector data management will be described. In section 4, we introduce the design and performance test of the paralleled detector simulation software. Finally, section 5 summarizes the current status and gives an outlook on the development of OSCAR.

2 Underlying framework

The OSCAR system is implemented based on SNiPER [9] as the underlying framework, which implements fundamental functionalities such as the event loop control, job configuration, multi-thread support, logging, job configuration etc. SNiPER also provides interfaces for developers to implement specific algorithms and services.

SNiPER is a lightweight software framework, initially developed and optimized for non-collider experiments (such as JUNO [10] and LHAASO [11]), and was then adopted to collider experiments as well. The main advantages provided by SNiPER include its lightweight design and implementation, flexible event processing sequence control, flexible data management and the powerful parallel computing support. In SNiPER, the basic execution unit is named as task, as shown in Figure 2 (a). A SNiPER task is a thin wrapper of the algorithms and services, where developers can plug-in their code to fulfill data processing tasks. To enhance SNiPER's flexibility, in each SNiPER job multiple task instances could be created,

each with a different sets of algorithms and services, and with an independent input/output stream. In each SNiPER job, users could define a top-level task which takes the primary event loop, and trigger other tasks on demand, as shown in Figure 2 (b). With such features, complex applications can be easily implemented, such as mixing different kinds of backgrounds with asynchronous event rates, triggering different processing chains for different kinds of physics signals, or analysing data coming from multiple input streams etc.

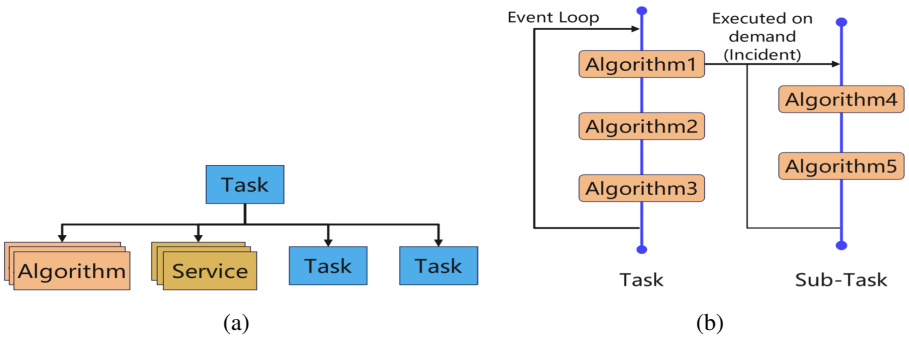


Figure 2: (a) Design of SNiPER Task. (b) Nested execution sequence of Tasks.

SNiPER also takes advantage of the multi-task mechanism to implement event-level multithreading applications based on Intel TBB [12]. In the multithreading mode of execution, the ordinary event loop is broken up by the SNiPER Muster (Multiple SNiPER Task Scheduler) [13], which uses the Intel's TBB's scheduler. As shown in Figure 3, the Muster works as a thread pool, and is in charge of the creation and scheduling of multiple workers. In each worker, one SNiPER task instance is created and mapped to an Intel TBB task, executed in one thread concurrently. When a task is invoked, it grabs and locks an event from the central event pool (GlobalStore) until the execution of the task holding this event is completed.

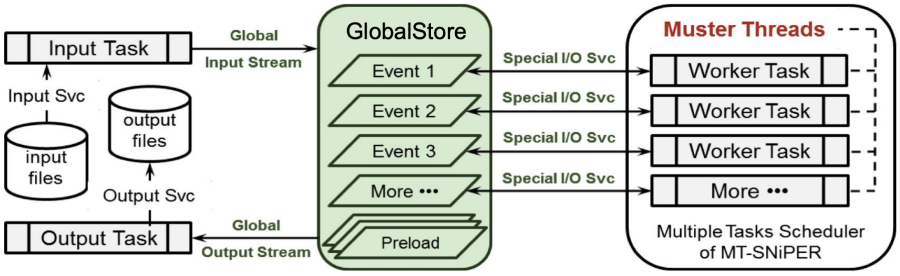


Figure 3: The concurrent execution of the SNiPER job.

To simplify the implementation of data I/O and memory management, the I/O services are configured in dedicated I/O tasks (threads). During execution, the input task always try to fill the GlobalStore until there is no empty slot, while the output task always try to write out processed events. Such design is in particular useful for experiments that require the sequence the event data to remain fixed all the time during offline processing (e.g. neutrino experiments that have time-correlated triggers). The memory management functionalities is

achieved by the GlobalStore, which is designed based on podio::EventStore, and is capable of caching multiple events during execution. The implementation details of GlobalStore will be discussed in the next section.

3 Event data and detector data management

3.1 Event data management

The EDM lies at the heart of the OSCAR system. It not only defines the transient and persistent format of event data, but also defines the relationships between event objects in different processing stages, and provides interfaces for various application algorithms during the offline processing.

The traditional EDM for HEP experiments has been predominantly object-oriented with complicated hierarchical structures. While these designs have effectively encapsulated event information and played crucial roles in the past, they are not well suited for future experiments with higher data rates and stronger demand for paralleled data processing. To implement effective EDM and data I/O for future HEP experiments, the Key4hep project has developed podio that serves as a common tool to define EDM based on plain-old-data structures, offering highly efficient I/O performance as well as robust support for parallel computing. The EDM in the OSCAR system is developed based on podio, so that the event information and relationships between different event data objects are described in yaml files, which are then used to automatically generate C++ code for all EDM classes.

Currently, the EDM classes for MC simulation and reconstruction are defined for the STCF detector, as shown in Figure 4. The MCParticle and ReconstructedParticle are designed as the core index for MC and reconstructed data, respectively. Specific classes for the tracker system, electromagnetic calorimeter, PID detectors and muon detectors are defined. The arrows in the figure denote the "one to one" or "one to many" relationship between these classes. The MCParticle and ReconstructedParticle classes are correlated based on a track matching algorithm, bridging the MC and reconstructed data.

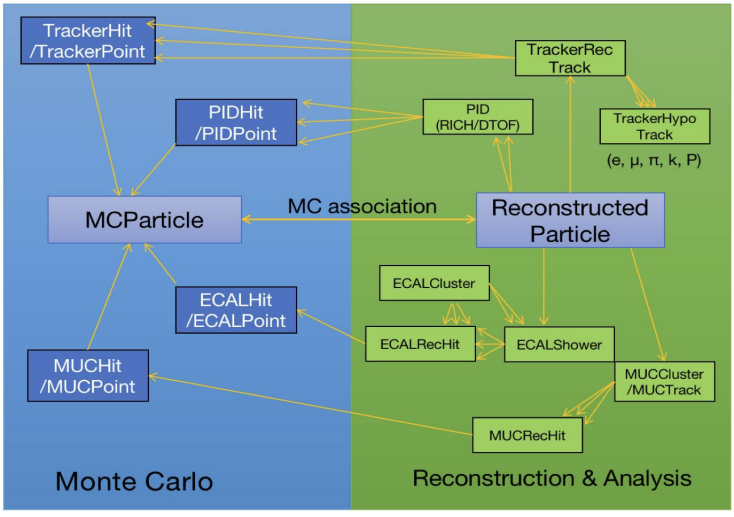


Figure 4: The design of OSCAR EDM.

The event data management system manages event data in memory (transient EDM objects and collections), provides interfaces for user applications and handles the data I/O. As both the SNiPER framework and podio have independent implementation of the data management functionalities, in OSCAR we integrate them by implementing a few services, including PodioDataSvc that connects the podio::EventStore with SNiPER, PodioInputSvc and PodioOutputSvc that connect podio::ROOTReader and podio::ROOTWriter as well as the DataHandle serving as the interface for retrieving event data. Figure 5 shows how these components work together with the user algorithms. Under such design, the event data and user application are completely decoupled.

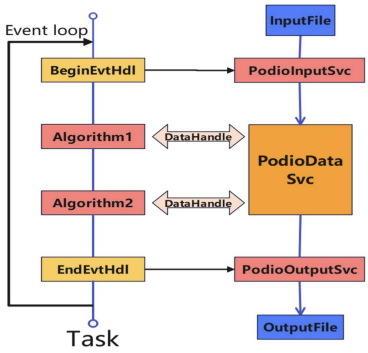


Figure 5: The design of OSCAR event data management

One of the biggest challenges of the event data management system is to fullfill the parallel computing requirements. To enable parallel data processing, the GlobalStore (as shown in Figure 6) is developed. The GlobalStore is designed based on the podio::EventStore to cache multiple events, each within one data slot. To guarantee thread safety, several condition locks are implemented to enable exchanging data safely between workers. As introduced in the previous section, I/O services are bound to dedicated I/O threads, to ensure performance and flexible post- or pre-processing. During concurrent execution, the input thread always tries to fill the GlobalStore, the output thread always tries to write out processed events, while each worker retrieves and locks one event each time before it is processed.

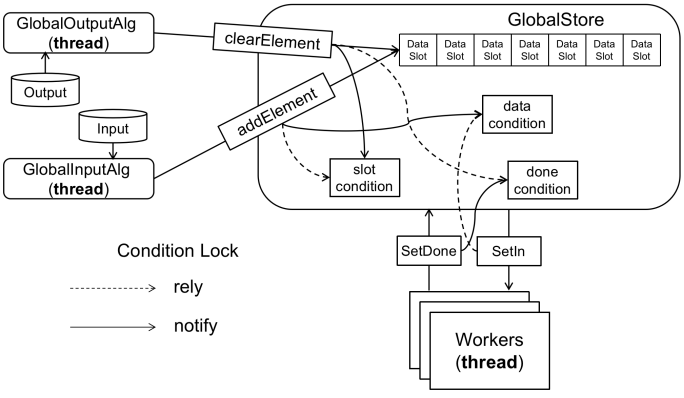
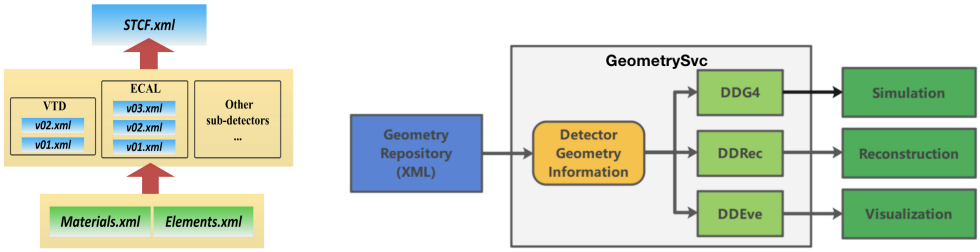


Figure 6: The design of GlobalStore

3.2 Detector data management

The detector data, including the detector geometry, material, detector readout as well as detector conditions data such as detector alignment, calibration parameters are vital during the entire offline data processing. To effectively manage these detector data and provide straightforward interfaces for applications, the geometry management system (GMS) [14] is developed, as one of the core components of the OSCAR system. To make sure all applications, including the detector, digitization, calibration, reconstruction and event visualization share consistency geometry definitions, the GMS is developed based on DD4hep that uses a single source of all detector data. In GMS, the definition of basic elements and materials is shared, while each sub-detector is defined in a single XML file. Each sub-detector can have an independent version number, and the full detector is defined in a mother XML file as a composition, in order to support flexible switch between different detector design schemes.



(a) (b)

Figure 7: (a) Detectors repository in GMS. (b) Data flow of geometry information managed with GeometrySvc.

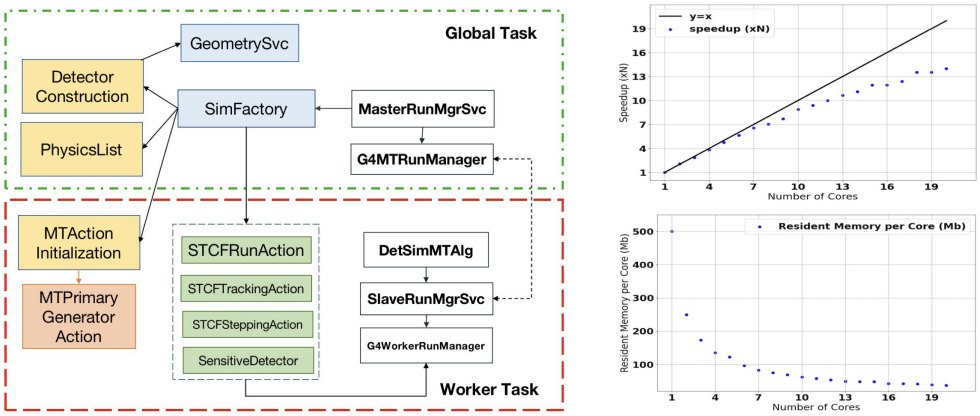
To integrate DD4hep with SNiPER, the GeometrySvc is developed as a manager of all detector data. During initialization, the detector geometry defined in XML files will be parsed and delivered via various plugins of DD4hep on demand. For instance, geometry is delivered to DDG4 to transfer the geometry to Geant4, to DDRec for the reconstruction algorithms, and to DDEve for detector and event visualization. With such implementations, the entire STCF detector is implemented to support current detector simulation and reconstruction algorithms.

4 Detector simulation framework

Generating MC simulated data is one of the primary tasks of OSCAR. To integrate SNiPER, Geant4 and the GMS, a modularised detector simulation framework is implemented for developers to flexibly develop modules such as generator interfaces, physics interaction lists, user-defined actions as well as fast simulation models.

The design of the detector simulation framework is shown in Figure 8 (a). In order to implement inter-event parallelism, the Geant4 related components are divided into two categories. The DetectorConstruction that constructs the detector geometry via the GeometrySvc and the PhysicsList are created in the global task, which are shared by all workers. The physics generator, sensitive detectors as well as user-defined action code are possessed by each worker. Under such design, users can run detector simulation serially or concurrently by just changing job configurations. Figure 8 (b) shows the result of some performance tests, obtained by concurrently simulating muon particles. By comparing the relative speed-up and

averaged memory consumption, we can clear see the system works as expected and shows a good scale-up capability.



(a) (b)

Figure 8: (a) Design of the parallel detector simulation framework. (b) Performance test of paralleled detector simulation.

5 Summary

In this paper, we introduce the design and implementation of the OSCAR system, the offline data processing software of the STCF experiment. OSCAR is developed fully based on SNiPER and partially based on Key4hep. To build efficient and powerful event data and detector data management systems, several tools developed for future collider experiments are adopted, such as podio and DD4hep. In the past years, the full detector simulation and reconstruction chain were built on top of OSCAR, and the latest update has enabled efficient paralleled detector simulation software. OSCAR is still under rapid development to enhance the functionality and performance. We hope the design of OSCAR can potentially provide solutions for other lightweight HEP experiments.

6 Acknowledgements

This work was supported by National Natural Science Foundation of China (NSFC) under Contracts Nos. 12025502, 12105158, 12175124.

References

[1] H.P. Peng, Y.H. Zheng, X.R. Zhou, Physics **49**, 513 (2020)
[2] M. Achasov et al. (2023), `arXiv:2303.15790`
[3] W.H. Huang, H. Li, H. Zhou, T. Li, Q.Y. Li, X.T. Huang, JINST **18**, P03004 (2023), `arXiv:2211.03137`
[4] R. Brun, F. Rademakers, Nucl. Instrum. Meth. A **389**, 81 (1997)
[5] S. Agostinelli et al. (GEANT4), Nucl. Instrum. Meth. A **506**, 250 (2003)

[6] G. Ganis, C. Helsens, V. Völkl, Eur. Phys. J. Plus **137**, 149 (2022), arXiv:2111.09874
[7] F. Gaede, G. Ganis, B. Hegner, C. Helsens, T. Madlener, A. Sailer, G.A. Stewart, V. Volkl, J. Wang, EPJ Web Conf. **251**, 03026 (2021)
[8] F. Gaede, M. Frank, M. Petric, A. Sailer, EPJ Web Conf. **245**, 02004 (2020)
[9] J.H. Zou, X.T. Huang, W.D. Li, T. Lin, T. Li, K. Zhang, Z.Y. Deng, G.F. Cao, J. Phys. Conf. Ser. **664**, 072053 (2015)
[10] Z. Djurcic et al. (JUNO) (2015), arXiv:1508.07166
[11] X.H. Ma, Y.J. Bi, Z. Cao, M.J. Chen, S.Z. Chen, Y.D. Cheng, G.H. Gong, M.H. Gu, H.H. He, C. Hou et al., Chinese Physics C **46**, 030001 (2022)
[12] C. Pheatt, Journal of Computing Sciences in Colleges **23**, 298 (2008)
[13] J.H. Zou, T. Lin, W.D. Li, X.T. Huang, T. Li, Z.Y. Deng, G.F. Cao, Z.Y. You, J. Phys. Conf. Ser. **1085**, 032009 (2018)
[14] H. Li, W.H. Huang, D. Liu, Y. Song, M. Shao, X.T. Huang, JINST **16**, T04004 (2021)