Research on Distributed Data Acquisition Software for High-Frame-Rate Area Detectors

Authors: Shang Kunlin, Zhengheng Li, Ju Xudong, Liu Zhi, Zhou Yue, Huai Ping, Li Zhengheng

Date: 2024-03-28T00:00:00+00:00

Abstract

High-frame-rate area detectors are key equipment for conducting scientific experiments such as coherent diffraction imaging and serial crystallography at the Shanghai Hard X-ray Free Electron Laser Facility (Shanghai HIgh repetitioN rate xfel and Extreme light facility, SHINE). The high-speed transmission and processing of massive data generated by area detectors poses significant challenges to data acquisition (DAQ) software. To meet the data transmission requirements of area detectors with megapixel scale and 10 kHz frame rate, DAQ needs to achieve a stable data transmission capability of $ $20 GB · s-1. This study developed a distributed DAQ software based on C++, aiming to achieve high-speed parallel data readout to meet the data acquisition requirements of high-frame-rate area detectors. This study successfully achieved high-flux data transmission and event reconstruction capabilities of 20 GB · s-1 across 4 nodes, and implemented and tested data preprocessing functions such as data calibration and lossless compression, as well as the overall distributed operation of the software. This study can provide necessary support for ultra-high-flux data acquisition in relevant experiments employing area detectors.

Full Text

Preamble

Vol. XX, No. X, XXX 20XX
NUCLEAR TECHNIQUES ChinaXiv

Research on Distributed Data Acquisition Software for High Frame Rate Area Detectors

SHANG Kunlin[1], LI Zhengheng[1], JU Xudong[1], ZHOU Yue[1], HUAI Ping[1,2], LIU Zhi[1]
[1](ShanghaiTech University, Shanghai 201210, China)

[2](Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China)

Abstract

[**Background**]: High frame rate area detectors are critical equipment for conducting coherent diffraction imaging, serial crystallography, and other scientific experiments at the Shanghai HIgh repetitioN rate xfel and Extreme light facility (SHINE). The massive data volumes generated by these detectors at extremely high transmission rates pose grand challenges for data acquisition (DAQ) systems. [**Purpose**]: To support continuous operation of megapixel detectors at 10 kHz frame rates, the DAQ software must deliver stable data throughput of no less than 20 GB $\cdot$ s^{-1}. [**Methods**]: This research developed a C++-based distributed DAQ software designed to enable high-speed parallel data readout to meet the requirements of high frame rate area detectors. The study adopts a memory-based, MapReduce-like approach for online event reconstruction and evaluates the Bitshuffle+LZ4/ZSTD data compression algorithm. [**Results**]: The software successfully achieved 20 GB $\cdot$ s^{-1} high-throughput data transmission and event reconstruction capability using four nodes. It also implements and tests various functions including data calibration, lossless compression, and distributed operation of the entire software system. [**Conclusions**]: This research provides essential support for ultra-high-throughput data acquisition in related experiments using area detectors.

Keywords: Area detector; Data acquisition; Data compression; High repetition rate; Distributed computing

X-ray Free Electron Lasers (XFEL) represent the latest generation of advanced light source facilities, offering exceptionally superior photon performance. Taking the under-construction Shanghai Hard X-ray Free Electron Laser Facility (SHINE) as an example, it features ultra-high repetition rates (up to 1 MHz), ultra-high brightness ($ 10^{12}$ photons/pulse), ultra-short pulses (~100 femtoseconds), and excellent coherence. Internationally, many technologically advanced nations including the United States, Germany, Japan, and Switzerland are actively engaged in XFEL research and construction. XFEL finds extensive applications across physics, chemistry, materials science, and life sciences [**?**]. For instance, Serial Femtosecond Crystallography (SFX) based on XFEL enables protein crystal structure determination at room temperature without cryogenic freezing. However, SFX requires collecting numerous diffraction patterns within short timeframes, imposing extremely high demands on high-frame-rate detectors and high-throughput data acquisition [**?**]. In 2018, the European XFEL (Eu-XFEL) experimentally demonstrated for the first time that SFX could deliver high-quality structural results at MHz-level X-ray pulse repetition rates [**?**], indicating that scientific experiments' demands for XFEL performance metrics such as repetition rate continue to grow.

Acquiring full-frame diffraction patterns at MHz-level rates in XFEL ex-

periments presents formidable challenges, requiring ultra-fast area detectors for image acquisition. Table 1 lists several area detectors used at XFEL facilities. Given that current XFEL area detectors remain immature and lack commercial products, SHINE has planned to independently develop a charge-integrating, ultra-high-frame-rate area detector named STARLIGHT (SemiconducTor Array detectoR with Large dynamIc ranGe and cHarge inTegrating readout). STARLIGHT aims to achieve frame refresh rates of $ 10*kHzandadynamicrangefromsingle* - *photonresolutionupto*10^{4}$ photons/pixel/pulse at 12 keV.

As XFEL pulse repetition rates and detector frame frequencies increase, scientific experimental data generation rates grow approximately exponentially, doubling every two years and expected to reach hundreds of GB $\cdot$ s^{-1} in the future [**?**]. Currently, the AGIPD detector' s processing software employs FPGAs for simple preprocessing and data forwarding, with further processing handled by servers [**?**]. The JUNGFRAU detector uses a single-server data reception and processing solution, achieving 9 GB $\cdot$ s^{-1} at 4M pixels and 1.1 kHz frame rate [**?**], and later 17 GB $\cdot$ s^{-1} at 4M pixels and 2.2 kHz frame rate [**?**] through FPGA/GPU acceleration within the server. However, compared with the rapid growth in data rates, computer hardware performance improvements lag significantly, making traditional single-server DAQ solutions increasingly inadequate [**?**]. The ePixHR detector' s data acquisition is implemented through the LCLS-II data system [**?**]. To handle this detector' s ultra-high data throughput, the LCLS-II system deploys a Data Reduction Pipeline (DRP) at the detector front-end, using hybrid FPGA/GPU/CPU architectures and multi-node distributed frameworks for compression and image filtering to reduce overall data throughput [**?**]. Evidently, to address growing area detector data throughput, combining intra-node FPGA/GPU heterogeneous acceleration with multi-node distributed scaling is necessary.

This paper addresses the challenges of area detector data acquisition and analysis, proposing a distributed software architecture tailored to the STARLIGHT detector' s requirements to enable high-speed reception and real-time processing of ultra-high-throughput data. The current study employs only CPU processors for data acquisition. The remaining sections are organized as follows: Section 1 introduces the area detector DAQ software requirements, Section 2 presents the software architecture, and subsequent sections cover implementation and testing.

1 STARLIGHT and Its Data Acquisition Requirements

Current megapixel area detectors used at XFEL facilities, such as CSPAD at LCLS, AGIPD at Eu-XFEL, and the under-development STARLIGHT at SHINE, all adopt modular designs [**?**]. A single STARLIGHT module contains 2$×8×128×128*pixels*, *achieves*10*kHzframerates*, *andisexpectedtogenerate*2*bytesperpixel*, *yieldingapproxim*
s^{-1}$ raw data rate per front-end module. STARLIGHT-1M comprises 4 modules (20 GB $\cdot$ s^{-1} total), while STARLIGHT-4M contains 16 mod-

ules (80 GB $\cdot$ s^{-1} total). The software performance target must at least meet STARLIGHT-1M specifications, while the architecture is designed for STARLIGHT-4M scalability. Figure 1 illustrates the STARLIGHT DAQ system hardware architecture.

To achieve high-throughput online data acquisition for STARLIGHT, the DAQ software must continuously receive raw experimental data from all detector modules at high speed. Data transmission channels between reception nodes and readout electronics must employ standard network protocols such as UDP or TCP based on performance and experimental requirements. Since area detectors acquire data through multiple front-end modules, each module transmits only partial image information, resulting in incomplete data at each reception node. This makes event selection and online analysis difficult, necessitating online event reconstruction to redistribute, aggregate, and assemble partial images from different nodes into complete images. To achieve high-throughput real-time storage, the software must provide real-time data compression capabilities. Additionally, to enhance user-friendliness, the software requires a graphical user interface (GUI) for real-time display of operational information and parameter adjustment.

2 Overall Software Architecture Design and Module Division

The software system comprises two main components: data acquisition/processing and front-end display. The data acquisition and processing software employs a distributed architecture inspired by MapReduce and Spark concepts, where multiple independent servers share data transmission and processing tasks. This design leverages multi-server computational and memory resources through pipelined parallelism, substantially increasing overall processing throughput [**?**]. Software modules operate independently, with state synchronization and real-time control implemented through a monitoring module. Inter-module command transmission and monitoring information exchange utilize WebSocket.

The front-end display software provides control and visualization interfaces, enabling real-time display of sampled image data. The overall architecture is shown in Figure 2. During data acquisition and processing, multiple Receiver modules directly acquire data from readout boards and distribute it to Combiner modules for event reconstruction. Event reconstruction aims to assemble partial images from different detector modules captured at the same time into complete images. The data then passes to processing modules for online calibration, compression, and storage.

The software employs ZeroMQ (ZMQ) as the high-throughput data streaming protocol between nodes, offering low latency, high throughput, and "zero-copy" transmission advantages suitable for high-speed bulk data transfer.

The DAQ software design consists of three primary modules: Data Reception

Module (Receiver), Event Reconstruction Module (Combiner), and Data Preprocessing Module (Processor), which includes data calibration and real-time compression.

2.1 Data Reception Module

The data reception module is a critical component of the detector DAQ software, directly connecting via Ethernet switches to each detector module' s backend readout electronics. Its primary responsibilities include data reception, buffering, and forwarding. Multiple data reception modules can be deployed across nodes for parallel data reception in high-throughput environments.

The transmission protocol between reception nodes and readout electronics can be selected from standard network protocols such as UDP or TCP. UDP is a connectionless protocol requiring no pre-established connection and minimal system resources. Although UDP may cause data loss, its implementation is relatively simple. TCP, by contrast, is a connection-oriented protocol establishing reliable connections.

Protocol selection depends on detector hardware configuration. Some detectors (e.g., JUNGFRAU) currently use UDP in their front-end electronics, where packet loss may occur. Lost packets can be handled by padding missing positions with blank data [**?**]. For experiments requiring high image integrity, such as X-ray Photon Correlation Spectroscopy (XPCS), data loss significantly impacts results. For the STARLIGHT detector under development, TCP is more appropriate given the high image integrity requirements.

2.2 Event Reconstruction Module

Image data event reconstruction relies on two key pieces of information: unified timestamp information within each detector module and the X-ray pulse number (Bunch ID) provided by the facility' s timing system. Event reconstruction merges partial image data with identical timestamps or Bunch IDs into complete images.

At Eu-XFEL, event reconstruction is performed before data enters the computing cluster using Train Builder electronic hardware, which reorganizes image data through multiple FPGA boards [**?**]. Referencing the data shuffle process in MapReduce, this paper presents a software-based event reconstruction solution using servers and switches [**?**], leveraging cross-node data exchange processes to fully utilize each node' s network bandwidth.

As shown in Figure 3, the event reconstruction module receives data distributed from all reception modules and merges partial images from different modules captured at the same time (based on Bunch ID or timestamp) into complete images (events). These complete images are then sent to processing modules for subsequent handling. For load balancing during operation, reception modules

can perform hash operations on Bunch IDs and take modulo the number of reconstruction modules to distribute data evenly, as shown in Equation 1.

The event reconstruction module employs multi-threading, where each thread receives data from corresponding reception modules and places it into partial image buffer queues for each detector module. It then merges module images with identical Bunch IDs into complete image data and places them into another queue for subsequent processing by the data processing module. To meet ultra-high-throughput processing demands, event reconstruction software is typically deployed distributed across multiple servers.

2.3 Data Preprocessing Module

To reduce overall data throughput and storage pressure, online preprocessing is required, including data calibration, compression/filtering, and formatted storage. This research focuses on online data calibration and compression algorithms.

Raw ADC data transmitted directly from detector modules contains non-uniformities such as pedestal (baseline) and gain variations across electronic channels. Data processing must subtract the baseline and correct gain non-uniformities to recover deposited energy or photon counts [**?**]. Following JUNGFRAU, AGIPD, and other area detectors, this correction can be expressed by Equation (2) [**?**], where ADU represents Analog-to-Digital Converter units:

$$energy[ADC[gain[]pedestal[ADUkeV$$

Based on Equation (2), the calibration module converts ADC data for all pixels in each image. It first reads calibration parameter data for each channel (including gain levels and corresponding pedestal parameters) from files into appropriate structures, then calculates deposited energy for each pixel according to Equation (2). Note that measuring detector pedestal, gain, and other calibration parameters requires dedicated research and experiments, which is beyond this paper' s scope. For software performance testing, this study currently uses simulated calibration parameters.

For data compression, JUNGFRAU employs the bitshuffle preprocessing method shown in Figure 4. This approach first rearranges data before applying compression algorithms. During experiments, most detector regions typically show weak signals, particularly areas far from the center. Consequently, diffraction images from area detectors exhibit relatively uniform and small pixel amplitude values in most regions, meaning the high bits of each 16-bit binary number are essentially zero. Leveraging this data characteristic, bitshuffle performs bit-level block transposition on image data, merging numerous high-order zero bits into complete bytes. This process does not change the original data size but significantly increases data redundancy. Combined with dictionary-based compression

relying on indexing, this method substantially improves compression efficiency. JUNGFRAU' s compression workflow consists of two steps: FPGA modules perform bitshuffle transposition, followed by software-based compression using general-purpose algorithms on CPUs [**?**].

This study tested the compression algorithm at the software level. The bitshuffle implementation used an open-source library [**?**], with subsequent compression performed using two common algorithm libraries: LZ4 and ZSTD [**?**].

3 Software Performance Testing and Optimization

Given that the STARLIGHT detector remains under development and cannot yet produce raw data, this study employs simulated experimental data for DAQ software performance testing. The simulation uses the publicly available CXIDB ID 83 dataset from Eu-XFEL' s femtosecond serial crystallography experiment with the AGIPD detector [**?**].

Table 2 shows the server cluster configurations for distributed and single-node performance tests. The distributed test cluster (Cluster A) contains eight servers: four for simulated data transmission and four for data reception and processing. To better evaluate software performance, server interconnections use 100G InfiniBand (IB) networks at the physical layer, with IPoIB modules enabling Ethernet-like data transmission at the software layer [**?**]. The single-node test cluster (Cluster B) contains five servers: four with large memory capacity for data transmission and one for reception and processing. Cluster B servers interconnect via a 25 Gbps Ethernet switch, with each of the four sending servers having 25 Gbps bandwidth and the receiving server having 100 Gbps bandwidth.

3.1 Data Transmission

For data transmission, this study conducted performance tests and comparisons of multiple methods on Cluster A, primarily testing TCP data transmission using BSD socket API, the boost::asio library, and thread CPU core binding, while also comparing synchronous and asynchronous modes in boost::asio.

As shown in Figure 5(a), performance using direct BSD socket API calls gradually increased with thread count but remained relatively slow overall, with four-thread synchronous reception reaching approximately 5.2 $\mathrm{GB \cdot s^{-1}}$. Performance improved after adopting the boost::asio library, particularly under multi-threading, achieving 7.3 $\mathrm{GB \cdot s^{-1}}$ with four threads. Figure 5(b) compares synchronous and asynchronous modes using boost::asio, showing synchronous transmission outperforming asynchronous. Figure 5(c) demonstrates that binding threads to fixed CPU cores further improved performance by reducing inter-thread competition and context switching, boosting four-thread synchronous reception to 8 $\mathrm{GB \cdot s^{-1}}$.

Performance tests indicate that boost::asio library usage yields superior trans-

mission rates compared to direct BSD socket API calls. Synchronous mode with boost::asio provides higher throughput, better suited for real-time high-speed reception of raw detector module data. Additionally, binding threads to fixed CPU cores reduces thread switching overhead and further enhances reception performance.

Tests were also conducted on the higher-performance Cluster B. With single CPU core binding and boost::asio library, synchronous TCP data transmission achieved 2.98 $GB \cdot s^{-1}$ single-thread reception rate, nearly reaching the data sending network card' s bandwidth limit (25 Gbps). Multi-threaded tests using four servers to simulate detector modules achieved 11.6 $GB \cdot s^{-1}$ with four threads, approaching the receiving network card' s bandwidth limit (100 Gbps).

As mentioned, a single STARLIGHT detector module' s raw data rate can reach 5 $GB \cdot s^{-1}$. Test results indicate this rate is difficult to achieve with a single thread but can be realized through two-thread concurrent reception, requiring the electronics data output to support multiple connections per port.

3.2 Event Reconstruction

This study conducted joint testing of the software' s data reception and event reconstruction modules, examining both single-node and multi-node scenarios.

Single-node testing: Using AGIPD detector simulation data generated from the CXIDB ID 83 dataset, 16 simulated data sender processes ran across four servers in Cluster B, continuously transmitting raw image data to the receiving server. Both reception and event reconstruction modules ran simultaneously on the receiving server for 30 minutes. During testing, total data transmission rate maintained 11.51 $GB \cdot s^{-1}$, reaching the network bandwidth limit while leaving over 30% CPU headroom. This demonstrates that online event reconstruction rate is no lower than data reception rate with low CPU resource utilization.

Multi-node testing: To evaluate software performance and stability in distributed multi-node environments, Cluster A was used to test data reception and event reconstruction modules. Four nodes simulated detector data transmission while another four performed reception and reconstruction, achieving approximately 23.5 $GB \cdot s^{-1}$ total parallel throughput. Network monitoring revealed reception node bandwidth was approximately 1.75 times the transmission node bandwidth. This occurs because event reconstruction and data reception nodes are deployed on the same servers, with both operations consuming the same network card bandwidth, resulting in higher total bandwidth usage than reception alone. Figure 6 shows single-node network card bandwidth occupancy during these tests. During event reconstruction, intra-node data exchange bypasses the network card, while each node must receive data distributed from the other three nodes. Combined with data flow from transmission nodes, the reception-to-transmission bandwidth ratio calculates as (3 + 4) / 4 = 1.75. Based on this analysis, single-node total data reception throughput reached approximately 10.3 $GB \cdot s^{-1}$, near the single-node network bandwidth limit,

explaining why the four-node reconstruction rate fell below the expected 40+ GB · s^{-1}. In actual deployment, data reception will use Ethernet while event reconstruction can use dedicated IB networks for intra-cluster data exchange, avoiding this bandwidth limitation.

3.3 Data Compression

Compression tests evaluated both compression rate and ratio using LZ4 and ZSTD methods on identical simulated data. ZSTD compression level is adjustable—higher levels yield slower compression but higher ratios. This study tested bitshuffle combined with ZSTD at compression levels 1, 5, and 10. Results are shown in Table 3 (with ZSTD levels in parentheses). Bitshuffle + LZ4 achieves faster compression speeds but lower ratios compared to ZSTD. Bitshuffle + ZSTD provides higher ratios but significantly lower speeds. Notably, using LZ4 alone produced larger files than the source, likely because LZ4 failed to compress the raw data effectively while adding dictionary index overhead.

A 2023 JUNGFRAU data processing paper reported achieving a compression ratio of 5.7 using FPGA-based bitshuffle processing combined with server-side LZ4 compression [**?**]. This study' s tests did not reach that maximum ratio, possibly because actual compression ratios depend on raw image data characteristics such as overall signal amplitude and per-pixel noise levels, which affect data redundancy after bitshuffle and thus compression efficiency.

3.4 Front-End Interface

The front-end interface module samples and displays images in real time, collects runtime information, and monitors software status. It also transmits runtime parameters—including configuration files—to other modules for online parameter adjustment, such as compression algorithm selection, compression level, data file format, and storage paths. Additionally, the module communicates with backend electronics to send slow control commands to the detector. This study conducted preliminary interface testing and development, with an example shown in Figure 7 displaying a complete AGIPD detector image after event reconstruction and calibration, using test data from CXIDB ID 83.

3.5 Overall Software Operation Testing

Overall software testing encompassed the complete pipeline: data transmission, event reconstruction, calibration, and compression. Both single-node and multi-node distributed tests were conducted using the same hardware configurations as the event reconstruction tests. In multi-node distributed testing on Cluster A, four sending servers each deployed four sender processes, continuously reading and transmitting data packets at 5,000 frames per second. Four receiving servers each deployed four receiver processes, one event reconstruction process, and four preprocessing processes. Average CPU usage during operation was approximately 70%, with overall data processing throughput around 8.6 GB ·

s^{-1}. In single-node testing on Cluster B, one receiving server deployed four preprocessing processes in addition to event reconstruction modules. CPU usage averaged approximately 85%, with overall processing throughput at 3.4 GB · s^{-1}. These results indicate that current servers struggle to meet tens of GB · s^{-1} processing requirements for area detectors. Stepwise testing revealed that data calibration, when saturating a single CPU core, only achieves 200 MB · s^{-1} processing rate, consuming excessive CPU resources in overall operation.

It is evident that CPU-only calibration and compression cannot meet requirements. While increasing server and CPU counts can improve overall throughput, handling tens or hundreds of GB · s^{-1} raw data streams would require prohibitively large server clusters. To control cluster scale and improve computational density, employing acceleration cards such as FPGA/GPUs for compute-intensive data processing will be essential.

4 Summary and Outlook

Addressing the challenges of ultra-high-throughput data acquisition and analysis for high frame rate area detectors, and to meet SHINE' s requirements for high transmission rates and preprocessing performance, this paper presents the development and testing of a series of distributed, high-performance DAQ and preprocessing software based on C++. The study first tested and compared data transmission methods, achieving 2.98 GB · s^{-1} single-thread TCP data reception using boost::asio in synchronous mode. A novel online event reconstruction method based on switches and server software was proposed, achieving approximately 11.51 GB · s^{-1} single-node raw data reception and online reconstruction rates, and approximately 23.5 GB · s^{-1} across four nodes. Test results demonstrate that the proposed software-based event reconstruction method operates with high speed and stability across multiple nodes, exhibiting good horizontal scalability. Additionally, this research developed and tested data calibration and lossless compression functions, evaluating bitshuffle+LZ4/ZSTD compression performance to establish a foundation for further detector DAQ software development and application.

Nevertheless, several aspects warrant optimization and improvement. Data calibration and compression currently use only CPU resources, consuming substantial CPU capacity and affecting overall software performance. To alleviate server burden, future work must leverage acceleration cards including FPGA/GPU for compute-intensive processing tasks.

Author Contributions

SHANG Kunlin developed software, conducted system testing and debugging, and wrote and revised the manuscript. HUAI Ping and LI Zhengheng designed the ultra-high frame rate, large dynamic range X-ray detection data acquisition system software, provided technical research guidance, and proposed and supervised research methodology. ZHOU Yue provided technical support and

participated in problem discussions. JU Xudong designed the detector scheme for the ultra-high frame rate, large dynamic range X-ray detection system and provided manuscript revision guidance. LIU Zhi, as project leader, provided research funding support and determined the overall scheme.

References

1. 赵振堂, 冯超. X 射线自由电子激光 [J]. 物理, 2018, 47(08): 481-490. Zhao Zhentang, Feng Chao. X-ray free electron laser [J]. Physics, 2018, 47(08): 481-490. DOI: 10.7693
2. 尹亮, 曾孟麒, 尹聪聪等. SHINE 束线站定时系统束团编号的数据采集 [J]. 核技术, 2023, 46(06): 060101. DOI: 10.11889/j.0253-3219.2023.hjs.46.060101.
3. Chapman, H., Fromme, P., Barty, A. et al. Femtosecond X-ray protein nanocrystallography. Nature 470, 73–77 (2011). DOI: 10.1038/nature09750
4. 千跃奇, 刘波. 串行晶体学数据筛选算法研究 [J]. 核技术, 2019, 42(08): 6-10. DOI: 10.11889/j.0253-3219.2019.hjs.42.080102.
5. Max O. Wiedorn, Dominik Oberthür, Richard Bean et al. Megahertz serial crystallography. Nature Communications, 9(1):4025, Oct 2018. DOI: 10.1038/s41467-018-06156-7
6. Filip Leonarski, Martin Brückner, Carlos Lopez-Cuenca et al. Jungfraujoch: hardware-accelerated data-acquisition system for kilohertz pixel-array X-ray detectors. J. Synchrotron Rad. (2023). 30, 227-234 DOI: 10.1107/S1600577522010268
7. Aschkan Allahgholi, J. Becker, A. Delfs, et al. The Adaptive Gain Integrating Pixel Detector at the European XFEL. Synchrotron Rad. (2019). 26, 74–82 DOI: 10.1107/S1600577518016077
8. Filip Leonarski, Aldo Mozzanica, Martin Brückner, et al. Jungfrau detector for brighter x-ray sources: Solutions for it and data science challenges in macromolecular crystallography. Struct Dyn. 2020 Feb 26;7(1):014305. DOI: 10.1063/1.5143480.
9. John L. Hennessy and David A. Patterson. 2019. A new golden age for computer architecture. Commun. ACM 62, 2 (February 2019), 48–60. DOI: 10.1145/3282307
10. J. B. Thayer, Gabriella Carini, Wilko Kroeger, et al., "Building a Data System for LCLS-II," 2017 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), Atlanta, GA, USA, 2017, pp. 1-4, DOI: 10.1109/NSSMIC.2017.8533033.
11. Ric Claus, Dan Damiani, Mikhail Dubrovin. et al. Data Reduction Activities at LCLS[R/OL]. (26-1-2024) [16-3-2024] https://www.xfel.eu/sites/sites_{custom}/site_{xfel}/conte 01-26DataReductionatLCLS_{eng}.pdf

12. Philip Hart, Sébastien Boutet, Gabriella Carini, et al. The CSPAD megapixel x-ray camera at LCLS. X-Ray Free-Electron Lasers: Beam Diagnostics, Beamline Instrumentation, and Applications, volume 8504, pages 51–61., 2012. DOI: 10.1117/12.930924

13. M. Zaharia, M. Chowdhury, T. Das, et al. "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 15-28, 2012. DOI:10.5555/2228298.2228301.

14. 卢晓旭, 顾旻皓, 朱科军等. LHAASO 地面簇射粒子阵列在线实时分布式数据处理 [J]. 核技术, 2020, 43(04): 76-84. DOI: 10.11889/j.0253-3219.2020.hjs.43.040402.

15. Dhanya Thattil, Erik Fröjdh. Sls Detector Package: source code [CP/OL]. [12-3-2022] https://github.com/slsdetectorgroup/slsDetectorPackage.

16. J Coughlan, C Day, S Galagedera, and R Halsall. The train builder data acquisition system for the european-XFEL. Journal of Instrumentation, 6(11):C11029–C11029, nov 2011. DOI: 10.1088/1748-0221/6/11/C11029

17. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In OSDI' 04: Sixth Symposium on Operating System Design and Implementation, pages 137–150, San Francisco, CA, 2004 DOI: 10.1145/1327452.1327492

18. S. Redford, M. Andrä, R. Barten, et al. Operation and performance of the JUNGFRAU photon detector during first FEL and synchrotron experiments. Journal of Instrumentation,13(11):C11006–C11006, nov 2018. DOI: 10.1088/1748-0221/13/11/C11006

19. A. Allahgholi, J. Becker, L. Bianco, et al. AGIPD, a high dynamic range fast detector for the european XFEL. Journal of Instrumentation, C01023 jan 2015 DOI: 10.1088/1748-0221/10/01/C01023

20. Kiyoshi Masui, Richard Shaw, Takeshi Yamamuro. et al. Bitshuffle: source code [CP/OL]. [27-12-2022] https://github.com/kiyo-masui/bitshuffle.

21. Yann Collet, Przemyslaw Skibinski, Takayuki Matsuoka. et al. LZ4: source code [CP/OL]. [20-1-2023] https://lz4.github.io/lz4.

22. David Reiss, Will Bailey, Pieter De Baets. et al. Zstandard: source code [CP/OL]. [22-1-2023] https://facebook.github.io/zstd.

23. Anton Barty. CXIDB ID 83: Experimental data set [DS/OL]. [11-10-2022] https://cxidb.org/id-83.html.

24. R. E. Grant, M. J. Rashti and A. Afsahi, "An Analysis of QoS Provisioning for Sockets Direct Protocol vs. IPoIB over Modern InfiniBand Networks," 2008 International Conference on Parallel Processing - Workshops, Portland, OR, USA, 2008, 79-86, DOI: 10.1109/ICPP-W.2008.25.

Note: Figure translations are in progress. See original paper for figures.