ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

PROTOTYPE OF IMAGE ACQUISITION AND STORAGE SYSTEM FOR SHINE*

Huihui Lv, Huan Zhao[†], Danping Bai, Xiaomin Liu
Shanghai Advanced Research Institute, Chinese Academy of Sciences
201204 Shanghai, P.R. China

Abstract

Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE) is a quasi-continuous wave hard X-ray free electron laser facility, which is currently under construction. The image acquisition and storage system has been designed to handle a large quantity of image data generated by the beam and X-ray diagnostics system, the laser system, etc. A prototype system with Camera Link cameras has been developed to acquire and to reliably transport data at a throughput of 1000MB/sec. The image data are transferred through ZeroMQ protocol to the storage where the image data and the relevant metadata are archived and made available for user analysis. For high-speed frames of image data storage, optimized schema is identified by comparing and testing four schemas. The image data are written to HDF5 files and the metadata pertaining to the image are stored in NoSQL database. It could deliver up to 1.2GB/sec storage speed. The performances are also contrasted between a stand-alone server and the Lustre file system. And the Lustre could provide a better performance. Details of the image acquisition, transfer, and storage schemas will be described in the paper.

INTRODUCTION

Motivated by the successful operation of X-ray FEL facilities worldwide and the great breakthroughs in atomic, molecular, and optical physics, condensed matter physics, matter in extreme conditions, chemistry and biology, the first hard X-ray FEL light source in China, the so called Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE), is under construction. SHINE will utilize a photocathode electron gun combined with the superconducting Linac to produce 8 GeV FEL quality electron beams with 1 MHz repetition rate [1].

A myriad of image data will be generated by the beam monitor system, the optical diagnostics system and the laser system, providing the required parameters for the accelerator operation and physics research. In order to measure the laser accurately, CCD cameras in the optical diagnostic system capture images at high speed. In addition, the beam cross section measured by the seed laser system and the drive laser system provides the basis for the commissioning and adjusting the devices ' parameters. Thus the accelerator has need of an efficient data acquisition and storage framework to accommodate the high-speed frames of image data, which is of great value to engineers and physicists to identify errors, component deterioration, poor process optimization, etc. They can also be used for big data analysis to improve control system stability and efficiency, and reduce maintenance cost. We have designed a general image system which is less expensive, using regular commercial hardware. It could acquire, transmit, and store the images at the speed of 1000MB/sec. The relevant tools are also developed to retrieve and display images on real time. Details are described in the following sections.

* Work supported by Shanghai Municipal Science and Technology Major Project(Grant No. 2017SHZDZX02)
† zhaohuan@zjlab.org.cn

ARCHITECTURE

The whole system as shown in Fig. 1 can be divided into five functional modules, namely acquisition, transmission, storage, retrieval and online display.

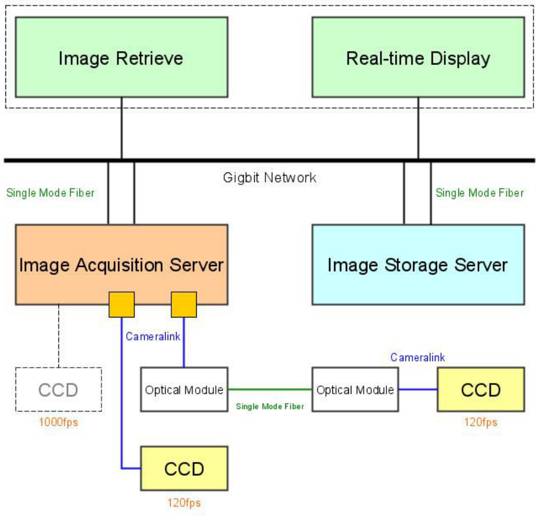


Figure 1: System Architecture.

Two CCD cameras with Camera Link interfaces take images at speed of 120 frames per second. One camera is directly connected to Camera Link Frame Grabber through a cable, while the other is connected to Camera Link Range Extender over a fiber cable, that solves distance limitation of Camera Link. Images are processed and packaged by the acquisition server and then transmitted to the storage server and online-display server through 10 Gigabit Ethernet, using ZeroMQ protocol for communication. After the storage server receives the data stream, it first unpacks it, then saves the image data in files as HDF5 format, and the metadata organized to facilitate searchability in MongoDB database. The web-based retrieval system is based on standard J2EE(Java 2 Platform, Enterprise Edition) Glassfish platform. It is designed to handle remote queries for historical records.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

The online display server can display the images captured from two cameras in real time.

Acquisition Module

There are three commonly used camera interfaces as shown in Table 1.

Table 1: Camera Interfaces Comparison

Interface	Performance
GigE	slower speeds, 100m cable length
Camera Link	5.4Gbps, short cable length, cost effective
CoaXPress	6.25Gbps, short cable length, expensive

GigE is usually used in systems with less critical speed and timing demands. Camera Link offers higher link speed up to 5.6 Gb/s (Gigabits per second), but cable length is limited and cable cost is a bit high. CoaXPress supports the transfer rate up to 6.25Gb/s, and it is more expensive than Camera Link. Taking cost and performance into consideration, we choose Camera Link and utilize a fiber optic range extender to resolve distance limitation by connecting camera to frame grabber over single fiber cable.

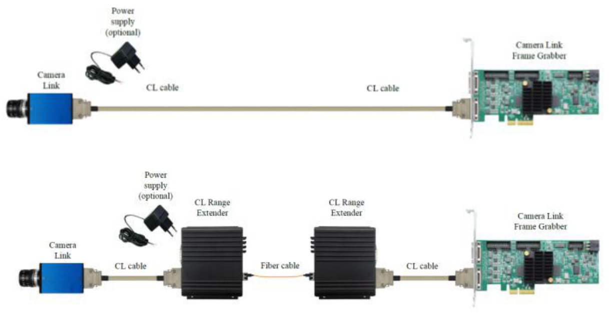


Figure 2: Camera Link Connection.

There are two cameras taking pictures at the same time. One is directly connected to Camera Link Frame Grabber through CL cable that is about 10 meters. The other is connected to CL Range Extender over single fiber cable that solves distance limitation of Camera Link. As shown in Fig. 2, the system is constructed of two converters, one on the camera side and one on the Camera Link frame grabber side. Camera Link Frame Grabber is a PCI Express-based device which allows high bandwidth communication between the device and the motherboard/server. The models of all devices are as shown in Table 2.

We use C language /Matrox Imaging Library(MIL) to develop programs for image capture, processing and annotation. MIL software development kit (SDK) is designed to reduce time and effort required to bring solutions. However, MIL SDK is mainly developed and utilized for Windows OS and has little support for Linux. It even cannot be installed in Linux. In order to install and use it in Linux, we tried multiple Linux versions, e.g., CentOS, Debian and Ubuntu. And finally we found Ubuntu 14.04 could match MATROX API.

Table 2: Devices List

Device Name	Performance	Number
CCD Camera	JAI SP-5000M-PMCL	2
Range Extender	Kaya Camralink Range Extender	2
Frame Grabber	Matrox_RAD EV 1G CLSF	2
CL cable		4
SFP module		2

Transmission Module

We make use of ZeroMQ[2] for image communication. ZeroMQ is a high-performance asynchronous messaging library, aimed at use in concurrent applications. ZeroMQ supports common messaging patterns (e.g., pub/sub, request/reply, client/server) over a variety of transports (e.g., TCP, in-process, inter-process, multicast and more), making inter-process messaging as simple as inter-thread messaging. We use pub/sub and request/reply patterns through TCP channels in the application.

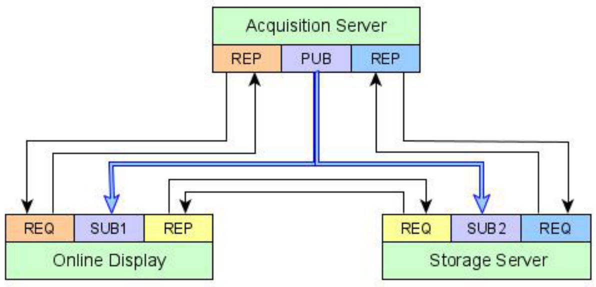


Figure 3: ZeroMQ Communication.

As shown in Fig. 3, request/reply connects the online display server (client) to the acquisition server (service), implementing the handshaking which should be successful before the transfer of data. Request/reply depends on a fixed send -> recv / recv ->send sequence. The connection will be interrupted if the sender does not receive the answer. Thus we modified two configuration parameters to realize that even if the sender does not receive a reply, it can continue to ask until the handshake is successful. The other two connections of acquisition server/storage server and display server/storage server are the same. Every two servers must shake hands to ensure the transmission synchronously.

Publish/subscribe connects the acquisition server and display/storage server. The publish-subscribe pattern is used for one-to-many distribution of data from a single publisher to multiple subscribers in a fan out fashion. Unlike request/reply pattern, messages are sent directly to the two servers, without the knowledge of what or if any subscriber of that knowledge exists. We use pub/sub pattern to transmit the stream of image data, allowing two clients to consume the stream.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

In order to get high transmitting speed,we modify the parameters of the ten gigabit network switch and servers, making multiple parameters match each other. After the modification, the transmission speed increases from 500MB/sec to 600MB/sec. To further increase the speed, we bind two network interfaces into a single logical ‘bonded’ interface as shown in Fig. 4. Network bonding increases the network throughout and bandwidth. Whereas network bonding increases CPU consumption and reduces overall performance. So we use two networks logically and in our program, data are transmitted through two networks at the same time. The total transmission speed increases to 1200MB/sec. That could satisfy our need.

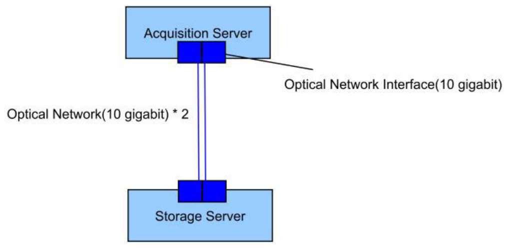

Acquisition Server
Optical Network Interface(10 gigabit)
Optical Network(10 gigabit) * 2
Storage Server

Figure 4: Network Bonding.

Storage Module

The image data represents an image, while the metadata pertaining to the image includes details relevant to the image itself as well as information about its production. What ' s more, the metadata is organized to facilitate searchability [3].

The image metadata involves:

- Number of the camera capturing the image
- Timestamp to the millisecond
- IP address
- Port number
- The camera resolution
- The depth of every pixels of the camera
- The gain of the camera
- The exposure time

We have designed four schemas for storage.

Storage Schema 1

The image data itself and the metadata are both stored in MongoDB. The image data is stored as a 2-D array of 8-bit unsigned integers.

Storage Schema 2

We make use of HDF5 as the permanent storage to store the image data, while using MongoDB as the index to store the metadata. The path where the image is stored is managed by MongoDB and used as the index to retrieve the HDF5 file.

Storage Schema 3

MongoDB is replaced by Cassandra.The metadata is stored in Cassandra as the index and the image data is stored as HDF5 format.

Storage Schema 4

Similar to Schema 2, we use DIRECT CHUNK WRITE strategy to write the data to HDF5 files. Others are same.

The storage performance is tested using three different sizes of gray level image, 1024×1024 bytes (1 MB),1024 × 2048 bytes (2 MB) and 2048 × 2048 bytes (4 MB), respectively. For each image, we record the time consumed to store 100 frames, 200 frames, 300 frames, 400 frames and 500 frames of images. The testing is performed in a rack-mounted server with 512 GB of RAM and 960 GB of SSD hard disk, running Linux/CentOS 7 operating system. There are 2 physical CPUs and each CPU has 6 cores in the server.

The test program is written to perform the same store operation 50 times constantly. Then we calculate the storage speed in MB/sec. The mean value with error bars is displayed in Fig. 5. The number of frames is on the x axis. The y axis corresponds to the storage speed in MB/sec. The image size is given on the bar. For instance, 1 M represents the image of 1024 × 1024 pixels (width: 1024, height: 1024) and each pixel is represented by a byte (8 bits).

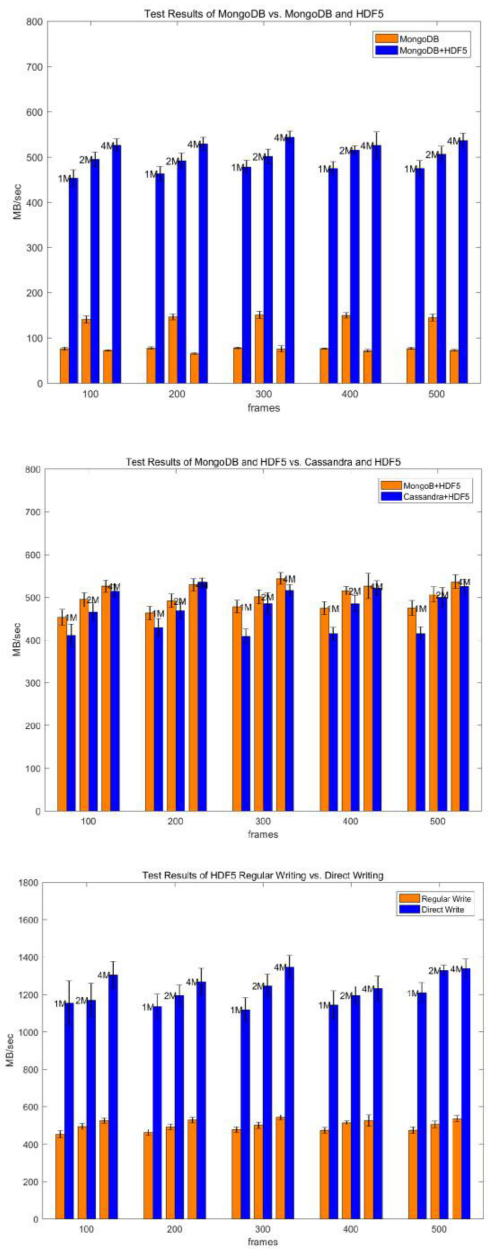


Figure 5:Test Results of Four Storage Schemas.

As we can derive from Fig. 5, to use MongoDB to store the metadata and HDF5 DIRECT CHUNK WRITE to store image data provides the optimal storage performance, about 1200MB/sec. It seems to be difficult to increase the speed from the software design. So as to achieve a higher storage speed, we can consider to extend the hardware. However, the cluster provided by several servers could deliver a combined throughput. If we stream the image data to a cluster, it is likely to further improve the storage speed. Consequently, we use three servers to deploy a Lustre file system. There are a single metadata target (MDT) and two object storage targets (OST). MDT holds metadata information, which is stored separately from file object data. The data content is written to OSTs to persistent storage. Meanwhile, we configure two clients mounting to the Lustre file system using the 10 gigabit network protocol. The image storage performance is tested in the cluster. And we calculate the sum of the storage speed from two storage processes. It is about 1500MB/sec. The result shows that the cluster provides more optimal performance than the stand-alone server. While the Lustre in the test is a small scale distributed file system, which is composed of three servers mounted by two clients. The Lustre system can scale to large scale platforms provided by more servers, and present larger capacity storage space to more clients. Then we can obtain superior combined throughput performance. If there is higher requirement of data storage speed, the Lustre file system could scale easily in the future.

Retrieval Module

The retrieval system is based on Maven J2EE Glassfish platform with MongoDB and HDF5 utilized as backend data storage. As shown in Fig. 6, the application architecture diagram is composed by three parts: the persistence layer, the business logic layer and the client layer. The persistence layer is a general data storage container for both metadata and image data. The business logic refers to the processing demand to the data carried by the specific client. Data APIs are predefined database queries to facilitate application programming, which provides convenient access to the database. Data APIs use JDBC Connector to connect to MongoDB database to get a specific image's timestamp, cameroNo, index, etc. With index to HDF5 files, it uses Java HDF5 Interface (JHI5)[4] to access the image in HDF5 format. HTTPServlets process demand to the data carried by the web client and return data in JSON format. The client layer utilizes HTML5 Canvas to draw the pixel-level image, Bootstrap to style response websites and jQuery to handle the event handling and Ajax.

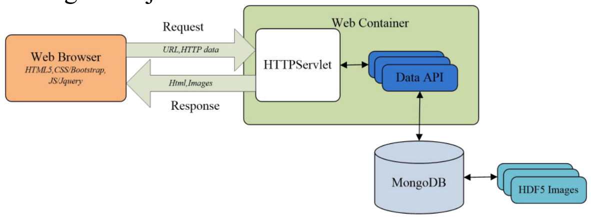


Figure 6: Retrieval Module Architecture.

Online Display Module

The online display module (Fig. 7) makes use of PyDM[5] to build graphic user interface. The image data transmitted through ZeroMQ protocol from the acquisition server are imported to the EPICS database located in the display server by PCASpy[6] interface. The total number of image frames acquired from the acquisition server is also displayed on this interface, compared with the number of images having been stored into the storage server (as shown in Fig. 8). The two are equal and it can indicate that there is no packet lost.

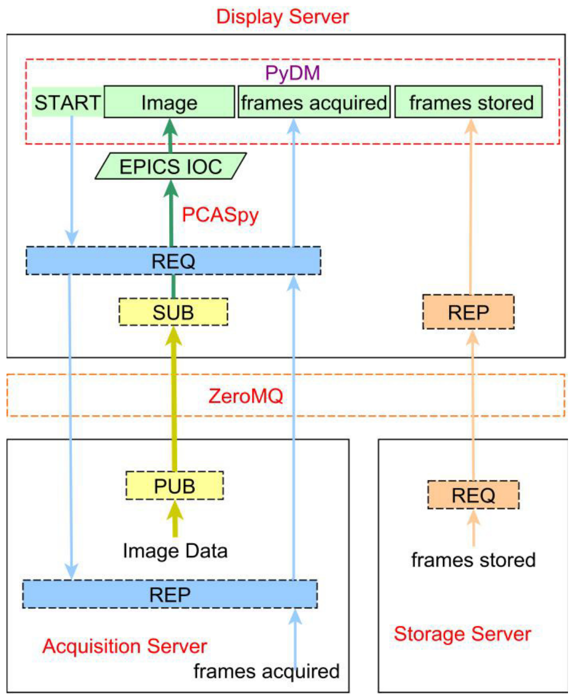
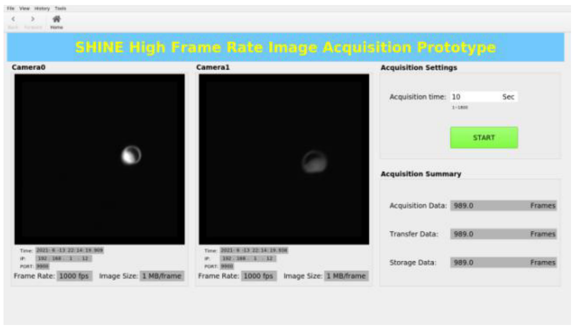


Figure 7: Online Display Module.



Acquisition time: 10 Sec

Acquisition Data: 989.0 Frames

Transfer Data: 989.0 Frames

Storage Data: 989.0 Frames

Frame Rate: 1000 fps Image Size: 1 MB/frame

Frame Rate: 1000 fps Image Size: 1 MB/frame

Figure 8: Online Display Interface.

TESTING

Test is taken by two cameras (120frames/sec, 1024*1024) illuminated by a diode which is excited by a sine wave signal generator as shown in Fig. 9. According to all images from one camera within two seconds, we find the brightest point of each image, the point with the largest gray value. Then we fit a sine curve using the least-squares method. The curve is shown in Fig. 10. The fitting frequency is equal to the frequency of the signal generator.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

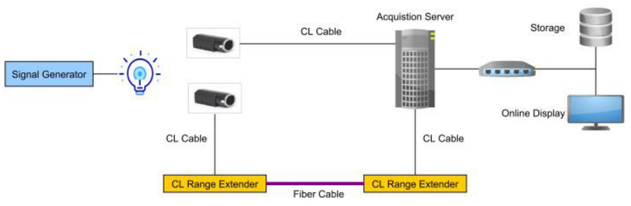


Figure 9: Testing.

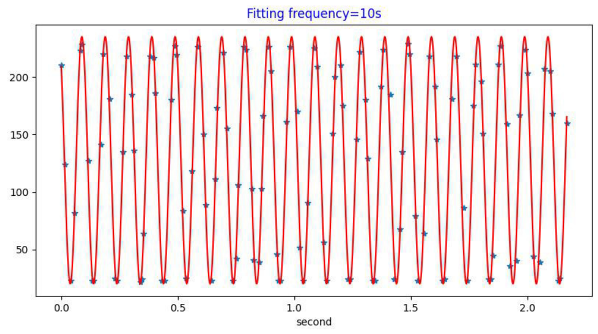


Figure 10: Fitting Curve.

We also test the performance of a higher speed, 1000 frames/sec with each image size 1MB. We monitor the number of HDF5 files generated every 5 seconds by Linux command ‘watch -n 5 "ls |wc -l |tee -a num.log"’. The result is shown in Fig. 11. Then the storage speed could be calculated, 500MB * 10 / 5sec = 1000 MB/sec. It shows that our system can get the speed of 1000MB/sec.

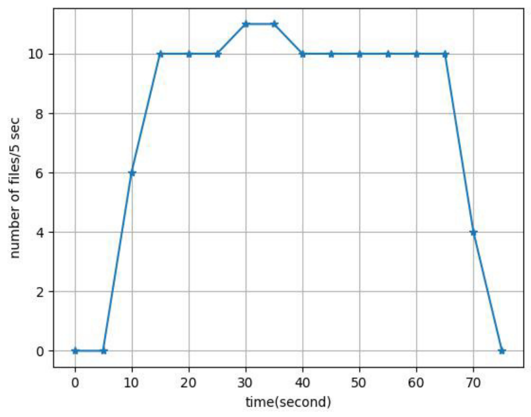


Figure 11: Test Results of 1000 Frames/sec.

CONCLUSION

The system is able to acquire, transmit and store the image data at speed of 1000MB/sec stably without loss. The solution is based on Camera Link interface to capture high-frame rate images, ZeroMQ protocol to transmit stream data, together with HDF5 to store the heavy image data and MongoDB for indexing the heavy data in HDF5. Multi-threading and network binding increases the holistic system performance as well. The hardware architecture and software design is not limited to image data. It could also manipulate the waveform data for SHINE.

REFERENCES

[1] Kai Li, Haixiao Deng, “Systematic design and three-dimensional simulation of X-ray FEL oscillator for Shanghai coherent light facility”, *Nucl. Instrum. Methods,* vol. 895, pp. 40–47, 2018. doi:10.1016/j.nima.2018.03.072

[2] https://zeromq.org/

[3] Lv H., Yan Y., Wang H., “The data storage system for SHINE”, *Nucl. Instrum. Methods,* vol. 1002, 2021. doi: 10.1016/J.NIMA.2021.165285

[4] https://www.hdfgroup.org/

[5] http://slaclab.github.io/pydm/

[6] https://pypi.org/project/pcaspy/