Development of High Level Trigger Software for Belle II at SuperKEKB

2011 J. Phys.: Conf. Ser. 331 022015

(http://iopscience.iop.org/1742-6596/331/2/022015)

Journal of Physics: Conference Series **331** (2011) 022015 doi:10.1088/1742-6596/331/2/022015

Development of High Level Trigger Software for Belle II at SuperKEKB

S Lee[1], R Itoh[2], N Katayama[2] and S Mineo[3]

[1] Department of Physics, Korea University, 1 Anam-dong, Seounbuk-gu, Republic of Korea
[2] IPNS, High Energy Accelerator Research Organization, 1-1 Oho, Tsukuba, Japan
[3] Department of Physics, University of Tokyo, 7-3-1 Hongo, Tokyo, Japan

E-mail: shlee@hep.korea.ac.kr

Abstract. The Belle collaboration has been trying for 10 years to reveal the mystery of the current matter-dominated universe. However, much more statistics is required to search for New Physics through quantum loops in decays of B mesons. In order to increase the experimental sensitivity, the next generation B-factory, SuperKEKB, is planned. The design luminosity of SuperKEKB is $8 \times 10^{35}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ a factor 40 above KEKB's peak luminosity. At this high luminosity, the level 1 trigger of the Belle II experiment will stream events of 300 kB size at a 30 kHz rate. To reduce the data flow to a manageable level, a high-level trigger (HLT) is needed, which will be implemented using the full offline reconstruction on a large scale PC farm. There, physics level event selection is performed, reducing the event rate by ~ 10 to a few kHz. To execute the reconstruction the HLT uses the offline event processing framework basf2, which has parallel processing capabilities used for multi-core processing and PC clusters. The event data handling in the HLT is totally object oriented utilizing ROOT I/O with a new method of object passing over the UNIX socket connection. Also under consideration is the use of the HLT output as well to reduce the pixel detector event size by only saving hits associated with a track, resulting in an additional data reduction of ~ 100 for the pixel detector. In this contribution, the design and implementation of the Belle II HLT are presented together with a report of preliminary testing results.

1. Introduction

The Belle experiment [1] has been performed for the last 10 years in order to solve puzzles of the current universe. It has observed CP violation [2] which supports the related theoretical model, the Kobayashi-Maskawa mechanism [3], that leaded to Nobel prize for Kobayashi and Maskawa in 2008. However, statistical uncertainties are still high so more statistics is required in order to test the current Standard Model precisely, to search for other sources of CP violation and to hunt for evidences of New Physics. The Belle experiment has been officially shut down and the Belle II experiment [4], the upgrade of Belle, is planned and approved. The target instantaneous luminosity of the Belle II experiment is $8 \times 10^{35}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ which is about 40 times larger than the one Belle achieved. Belle has accumulated about 1 ab^{-1} data in its 10 years operation and it is expected that the Belle II experiment will accumulate about 50 ab^{-1} data. Therefore, Belle II will be a great challenge in the aspect of computing.

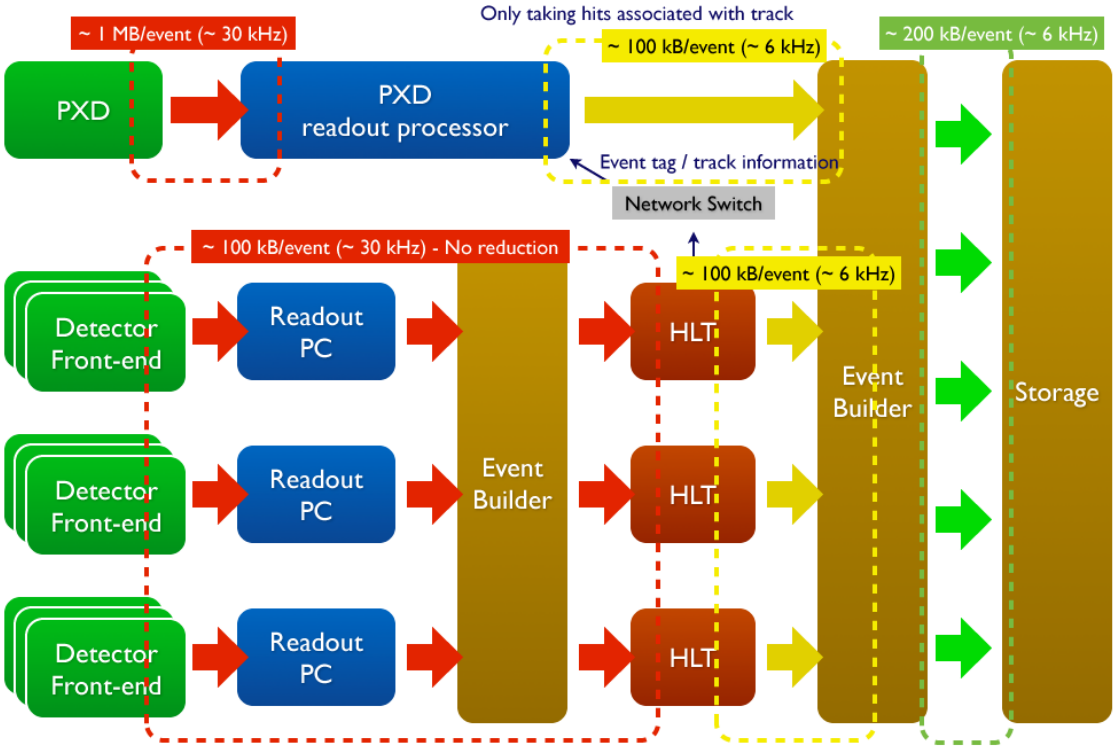

Figure 1. Simplified view of the Belle II data acquisition system

2. The Belle II Data Acquisition System

In order to take data from the experiment efficiently, data reduction will be performed within the Belle II data acquisition (DAQ) system as shown in figure 1. There are two data streams from the detectors. One is from the pixel detector (PXD) and the other one is from the other detectors without PXD. The data stream from PXD is processed by the PXD readout processor and the other stream is handled by high level trigger (HLT) system. The data size reduction occurs in the PXD readout processor while the data rate reductions occur in both the PXD readout processor and HLT. During the triggering, HLT gives the PXD readout processor some information such as event tag that includes which events are accepted by HLT, and tracking information that HLT reconstructed. PXD readout processor reduces not only the data rate by using event tag but also the data size by using track information. It only accepts hits which are associated with tracks reconstructed by HLT. In this triggering system, we expect the data rate reduction of factor 1/5 and data size reduction of factor 1/10. The HLT system of Belle II is only involved in data rate reduction and is located between two event builders which build event data.

3. The Belle II High Level Trigger System

The HLT system of Belle II is shown in figure 2. The entire HLT farm consists of about 10 units and each units contains about 20 nodes inside. There are three kinds of nodes in a single unit and they are the event separator, worker node, and event merger. The event separator distributes data to the worker nodes over the network, the worker nodes perform actual data reduction, and the event merger collects the data processed. There is also a special node called manager node which manages and monitors the entire HLT farm. On the worker nodes, a software framework works on the event processing. We developed a unified software framework

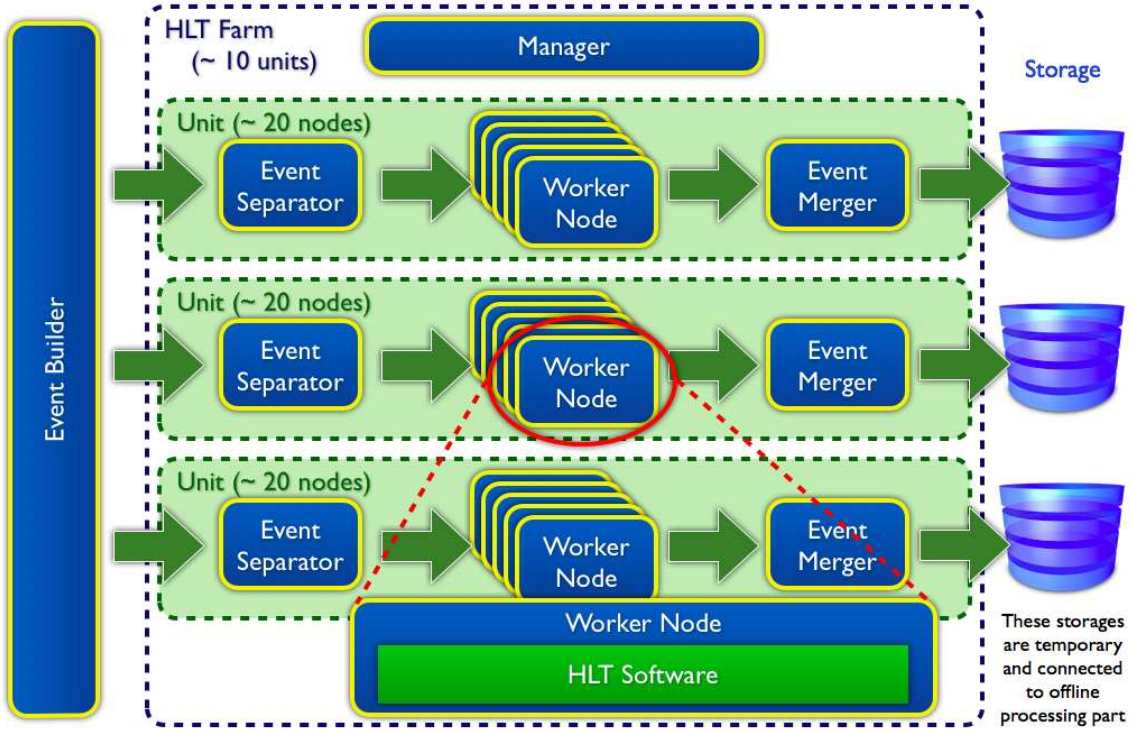

Figure 2. The Belle II high level trigger system. Every node runs HLT software that includes several components described in this paper.

named basf2 and intended that the same framework is used for both online and offline analyses. The framework has modular architecture so called software pipeline so that every analysis step can be regarded as modules. A module chain can be made by combining these modules so that the module chain actually performs the event processing. These modules can be attached and detached on demand so that the framework becomes very flexible. The framework is described in more detail in the article "The Software Framework of the Belle II Experiment" by Andreas Moll in these proceedings.

4. Data communication

The data transfer in the HLT system is based on the high speed local network so that the HLT software should take care of the network-based data communication. The data communication should guarantee the purity of the data so that we choose the TCP socket for data communication. The HLT software uses raw TCP socket so called UNIX socket. We developed a C++ socket class named B2Socket which provides basic functions to deal with socket communication. The B2Socket class has very low level interface so that we also implemented another classes named EvtSender and EvtReceiver. Those classes provide an interface to use B2Socket. As the names indicate, EvtSender is involved in sending data to one another and EvtReceiver is involved in retrieving data from others.

The data communication by EvtSender and EvtReceiver should be done independently so that those components should work as independent processes inside the operating system. In

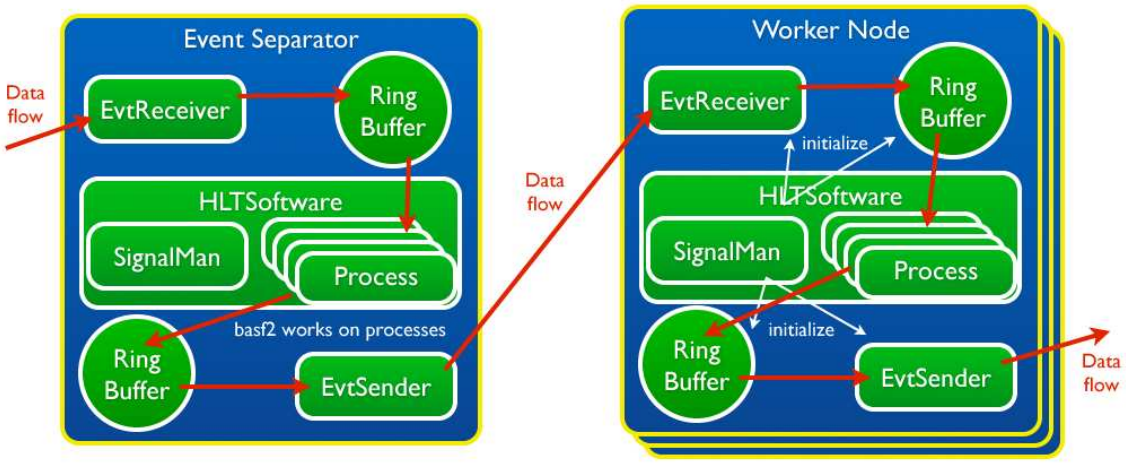

Figure 3. Data flow in the HLT system for the event separator and the worker nodes. Multiple boxes of process indicate multi-core capability. In the event separator, multi-core capability is shown as well as worker nodes but it doesn't need to be parallelized in real case. The software framework, basf2, is run in processes.

order to handle them, we implemented an additional component called SignalMan. This class not only forks and manages EvtSender and EvtReceiver but also initializes and manages shared memory-based ring buffers which imply semaphores [5] for synchronization for the interprocess communication among EvtSender, EvtReceiver, and the framework. Incoming data are accepted by EvtReceiver, stored into a ring buffer and passed to the framework. After the processing, the framework stores the processed data into another ring buffer and finally EvtSender sends the data processed to other nodes. This data flow is shown in figure 3. EvtSender and EvtReceiver work simultaneously, so enabling full-duplex communication.

5. Parallel Processing

In order to overcome the limit of capacity in event processing of a single CPU, parallel processing of events should be considered. We have two approaches to the parallel processing. One is multi-core based parallelization and the other one is multi-node based parallelization. The former is implemented as part of basf2 while the latter is implemented as part of HLT software.

5.1. Multi-core capability

For multi-core capability, we are using a forking mechanism. After the initialization of the framework, the framework forks off the main components related to multi-core based parallel processing. These components are event server, event process, and output server as shown in figure 4 and they are components of basf2. The event server separates data event by event and distributes them to the event processes. The event processes perform event processing through a module chain that is supposed to process event data. The output server collects and merges the processed data. These components are all independent processes so that there are shared memory based FIFO (First-In-First-Out) for interprocess communication. However, histogram parallelization is treated separately. Every event processes makes their own local histogram files during the processing and the framework collects and merges them into the one output after the event processing.

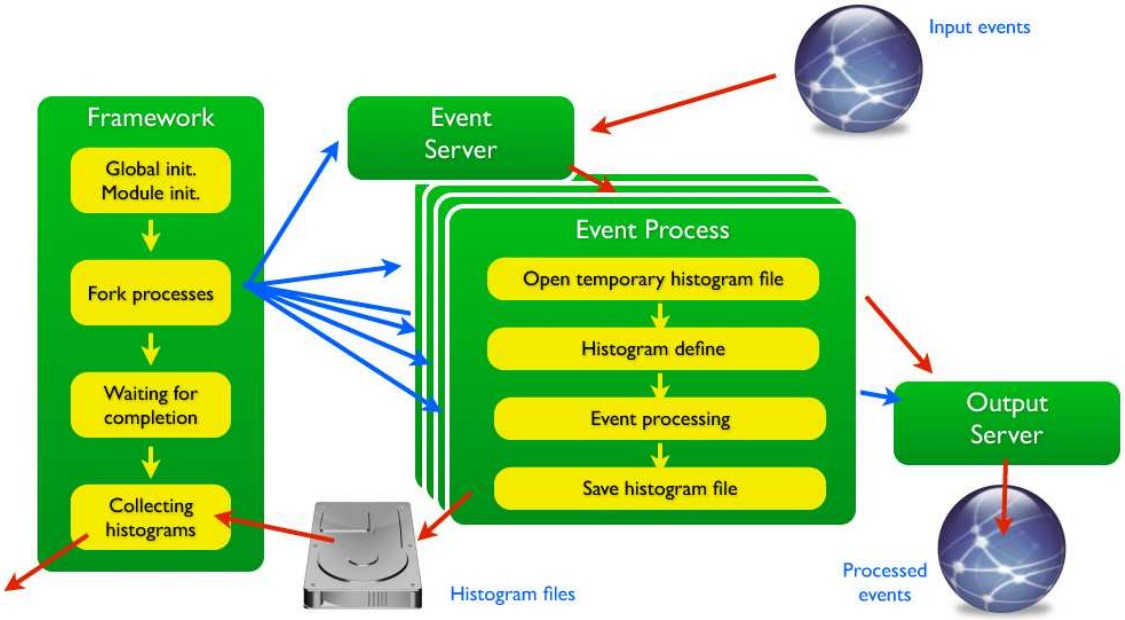

Figure 4. Multi-core capability of the framework. Event server, event process, and output server are components of basf2. This figure only describes a parallel processing in a single node so that networking is omitted here.

5.2. Multi-node capability

Another approach is multi-node based parallel processing. By combining multiple nodes, event processing can be parallelized over network. Therefore, data communication over the network is an important issue in this approach. As mentioned in previous section, we implemented B2Socket for the data communication. The data flow in multi-node based parallel processing is shown in figure 3.

6. Node management

Another important issue in the multi-node based parallel processing is a management system. As mentioned before, there is a special node called manager node (figure 2). The manager node has a hierarchy of components called HLTManager, UnitManager, and NodeManager. The HLTManager contains a number of the UnitManagers corresponding to the number of units inside the HLT system, and each UnitManager has a number of the NodeManagers corresponding the number of nodes in a single unit. On the node side except for the manager node, there are also the NodeManagers so that NodeManagers in the manager node and the corresponding node are paired to each other so that they exchange node information and monitoring information. For node information, a simple object called NodeInfo which contains an individual node information is serialized and transferred from the manager node to a node. This node information is provided as eXtensible Markup Language (XML) [6] format so that the input file is human readable, simple, and easily acceptable by other systems.

In this management system, we intend that all process nodes are initialized automatically. When the system starts, all nodes are initialized as a sort of abstract nodes, but they don't know what kind of nodes they are. After node initialization, all nodes wait for their own node information from the manager node. After broadcasting the node information from the manager node, the process nodes specify themselves by using the node information taken from the manager node, then, the process nodes can distinguish what kind of nodes they are. After

specifying all nodes, they start to work properly as their own roles. This automatic initialization and specification is done at the beginning of every run.

7. Current status

All related components of the HLT software has been designed and implemented although they need to be optimized for the realistic cases. The implemented components are tested separately whether they function correctly or not. For the test, we built a HLT test bench by combining a few multi-core built-in nodes. The integration of the HLT software is in progress and full functionality and performance test will be done taking into account the realistic cases. For more realistic test, we are planning to build a new test bench which is close to the HLT system for the Belle II experiment. We intend the HLT system is scalable so that we expect the performance will increase as the number of nodes increase. The scalability for multi-core capability has been studied in our previous study [7] and it will be studied for multi-node capability as well.

8. Summary

The high level trigger system for the Belle II experiment should be powerful, flexible, versatile, and scalable in order to take care of the huge amount of data expected in the Belle II experiment. We are developing a new high level trigger software which is based on the unified framework named basf2. The HLT software have parallel processing capabilities, multi-core capability and multi-node capability. By taking full advantage of these parallel processing, the performance of the HLT system will be much higher than the one of the previous experiment. In addition, the HLT software provides a flexible and automated management system so that it manages and monitors the entire HLT system effectively.

We have tested the basic functions of the HLT software on our own test bench. The HLT software is now under integration and we will test full functionality and performance on the new test bench which we will build soon.

Acknowledgments

Soohyung Lee acknowledges partial support from NSDC of KISTI.

References

[1] Abashian A *et al.* 2002 *Nucl. Instr. and Meth.* A**479** 117-232
[2] Abe K *et al.* 2001 *Phys. Rev. Lett.* **87** 091802
[3] Kobayashi M and Maskawa K 1973 *Progr. Theor. Phys.* **49** 652
[4] Abe T *et al.* 2010 Belle II Technical Design Report *Preprint* physics/1011.0352
[5] Djikstra W E 1967 *Proc. of the 1st ACM symp. on Operating System Principle* 10.1-10.6
[6] Bray T *et al.* 2008 Extensible Markup Language (XML) 1.0 (Fifth Edition) available from http://www.w3.org/TR/REC-xml
[7] Lee S. *et al.* 2010 *J. Phys.: Conf. Series* **219** 022012