Online and offline data handling for large detectors
(designed for non-accelerator experiments)

Yoshinari Hayato (ICRR)

- Introduction
- Difficulty (Water Cherenkov detector)
- Data transfer via Ethernet
- High Availability
- Data format for the online system
- In designing the data handling system
- Use of the database
- Summary

Introduction

Large neutrino and nucleon decay detector

Requirements for the online/DAQ system are
(slightly) different from the "accelerator" based experiment.

- Rare events (nucleon decay, super nova etc.)
  can happen "at any time".
- Natural neutrino beam is always there.
- Order of the events is also important.
  - Electron generated by decay of μ
    might be misidentified as the product of ν_e scattering.
  - Low energy events might be generated by spallation products

- Keep the detector running "**all the time**" without dead-time.
  Dead-time needs be minimized
  Dead-time should be recorded.
  1) Dead-time (down time) due to the equipment trouble or maintenance
  2) Dead-time due to the limitation of the DAQ/online system
- Triggered by the detector (system) itself.
- **Condition of the detector is necessary to be recorded.**

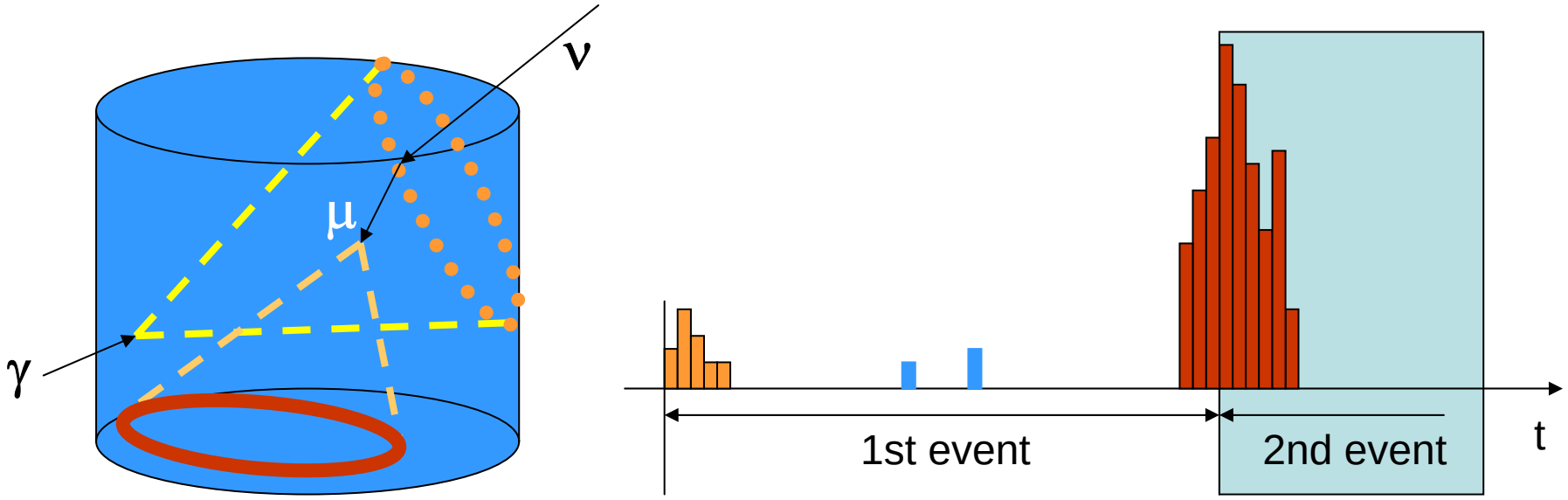
Difficulty

(Water Cherenkov detector and relatively extreme?? case)

If one want to detect very low energy events,
trigger rate gets extremely high.

(This depends on the radio activity in the water
and isolation from the surrounding rock.)

- Example of a very low energy event:
  2.2 MeV γ (from neutron capture) generates only ~6 PMT hits in SK.



Due to a trigger generated by a small activity,
"real high energy ν event" can be split into two.

Simple solution

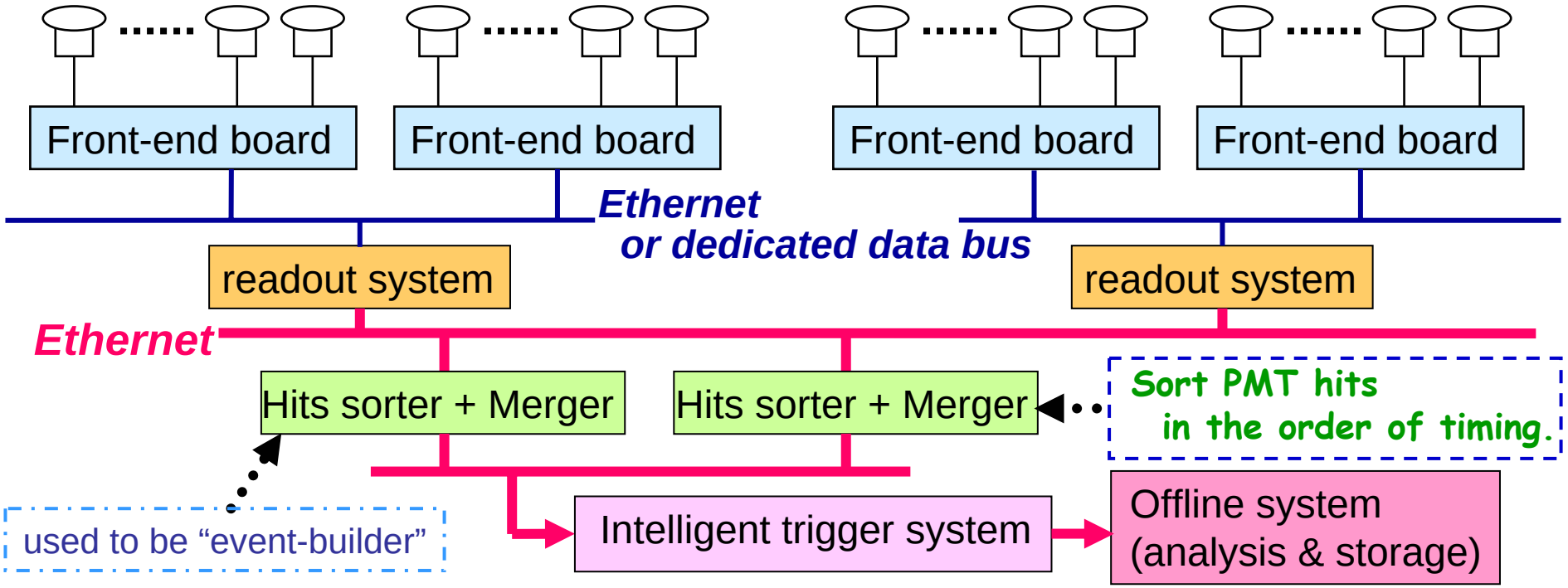
Record all the hits from PMTs including dark noise.

Then, apply the "software" trigger and define an event.

Realistic??

Assume 10kHz dark rate and data size of 6bytes/PMT hit
~840MB/sec for a detector with 14k PMTs.
(Typical dark rate of PMT in SK is 3~5kHz.)

- **Possible design (Schematic diagram of data flow part)**



Data transfer via Ethernet

- Network performance heavily depends
  on the **size of each data packet**
  and the design of the **network topology**.
- The characteristic of the data flow of online/DAQ system
  is different from the usual client-server system.
  Usually, the server sends large amount of data
  based on the request from a client.
  In the DAQ system, server "collects" data.
  The network components are (sometimes) designed
  to maximize their performance for the "usual" client-server model.
- Network is quite reliable but not perfect, as always.
  Even TCP/IP might be a cause of problem.
  Device driver or network equipment might be broken or buggy.
  Some defects in equipment or software creates corrupted data.

➡ Careful design of the network is very important.

- Traffic analysis
  including typical packet (data block) size estimation
- Monitoring point (where and how to tap etc…)

High availability (HA) system

In the online/DAQ system

one of the computer will be down at any moment.

To reduce downtime of the detector,

concept of the HA system seems to be useful and attractive.

Always monitor the status of the system (computers).

Once one of the computer seems to have problem,

the stand-by machine starts working.

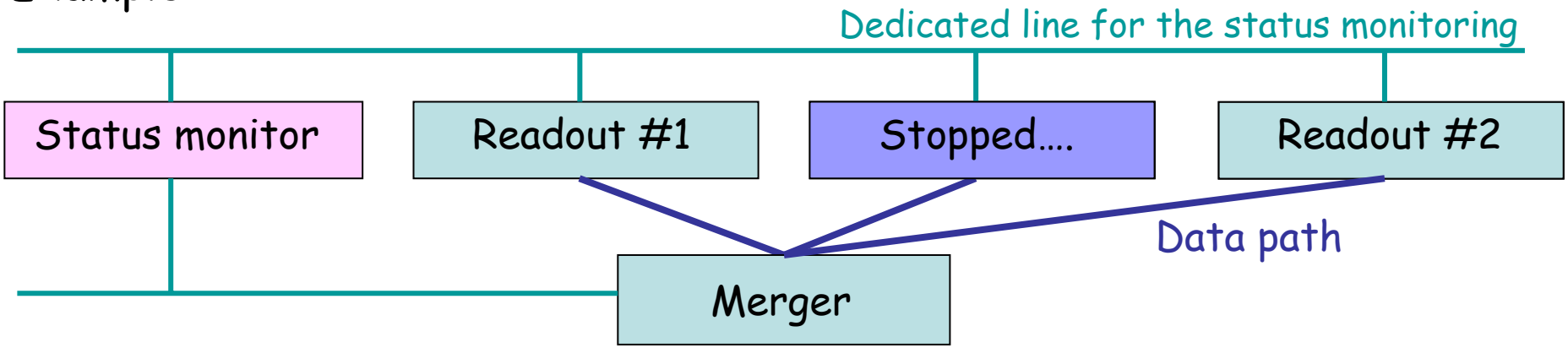
(Used for web, database and disk servers.)

Usually, complete replica of the server is prepared.

For the online/DAQ system, it is quite symmetric and thus,

only a few backups for each system will be sufficient.

Example



Data format used in the online system

Could be different from the one used in the offline system.

Preparing a data format converter
is easier and safer in various reasons, still.

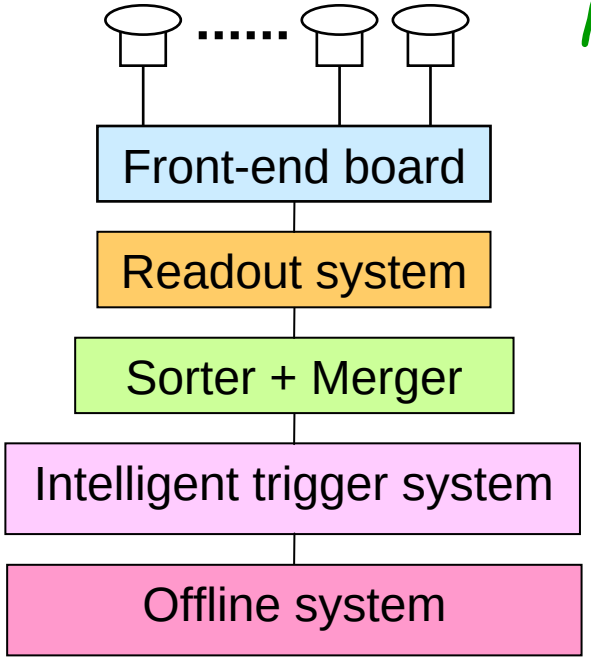
- Simple format and embedded "easy to identify" tags
  are always very useful.

  You may need to capture and analyze data on the network
  to investigate the problem.

- It is not easy to keep up with the updates of the offline software.
  - Once you start the experiment,
    people want to take data as much as possible.
  - But simple update of the library
    may cause "unexpected" troubles…

Also, it is helpful
to keep the "original" form of the (corrupted) data
in identifying the cause of a trouble in the offline format.

In designing the data handling system



Monitoring the data quality
and the quick detection of errors
are important issues

Traceability

- Necessary for the troubleshooting.
- You may want to analyze data with minor troubles.

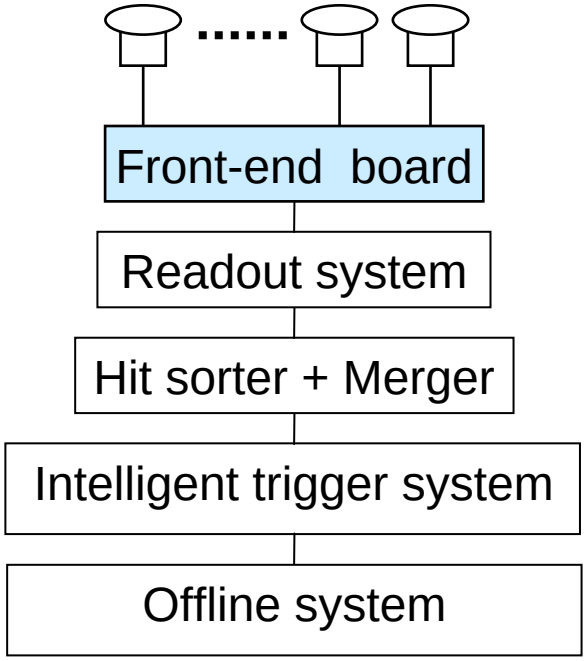
To understand the situation,
it is necessary to keep "erroneous" data.

Error detection - Possible check points:

1) Electronics (front-end board)
2) Readout system
3) Hits sorter + Merger
4) Intelligent trigger system
5) Data quality monitor

At each checkpoint, characteristics of accumulated data are different.
Monitor data quality at each point will be effective.

In designing the data handling system



1st check point

Front-end electronics board

One requirement for the electronics
(in the continuous data taking mode)

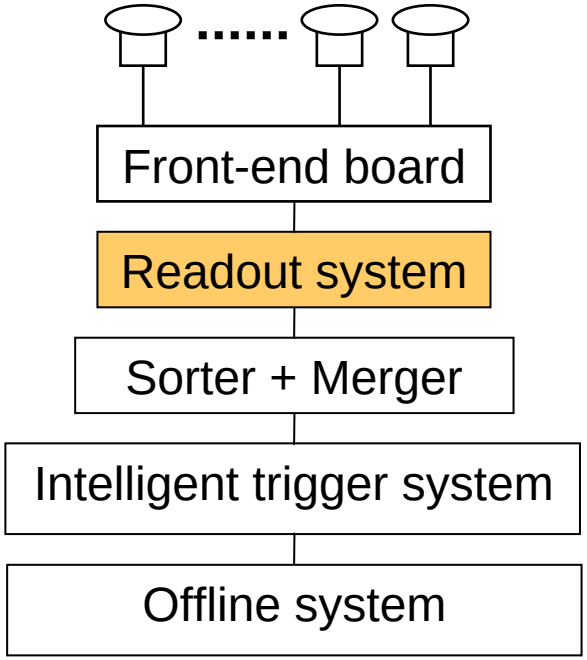
The front-end board should be designed
to generate periodic "heart beat" signal.

There might be no PMT hits in a board
for certain period of time.

It is difficult to identify
whether the board without PMT hits
was alive or not in the later check points.

Recent FPGA can provide smart hardware level error detection.
In preparing the error detection code,
don't forget to keep both the error information
and "erroneous data" themselves.

In designing the data handling system



2nd check point

Readout system

Readout system can be consists of
hardware modules only.
It will be robust in one sense.

But I (personally) think it is better
to have a computer in the system
at least for the error detection.

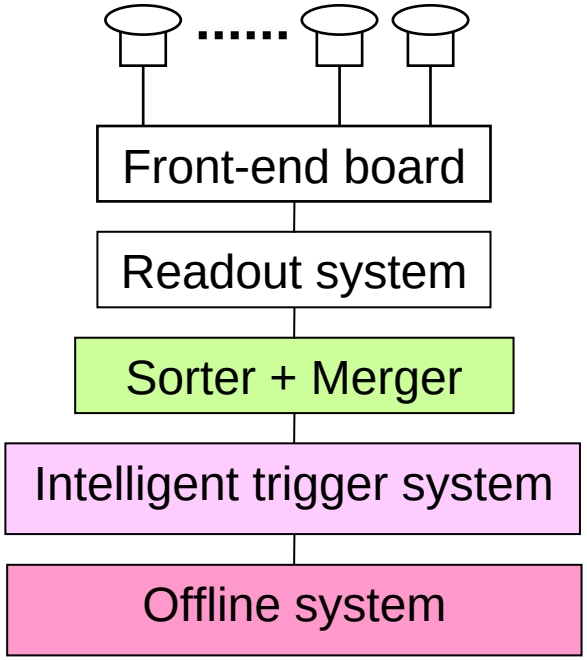
Because each readout system does not handle
not so many PMT hits.

And thus, it is possible to perform
detailed consistency check of the data here.

(Too much data makes you trouble in finding errors.)

Also, this computer will allow you
to test the system in the "standalone mode".
And you can run the remaining part of the detector.

In designing the data handling system



Next check points

Hit sorter + Merger

Intelligent trigger system

Offline analysis

Here, it is possible to use
all the hits from entire detector.

In applying the intelligent-trigger,
it is very important to keep
the revisions of the calibration constants
and software used to select (define) an event.

Same kind of information should be kept
in the offline analysis as always.

Also, the relation between the raw data (raw hits)
and the reconstructed results should be kept.

**Slow monitor information (low voltage/high voltage, temperature etc.)
are also very important.**

For example, missing block need to be "corrected" in the calorimeters.

Use of the database

(Personal opinion.)

As a (raw) data storage system,
there will be a problem in distributing data.

Data replication is possible but not so simple.
Supporting various data access method (direct file I/O & DB I/F)
might cause problem.

Of course, database is very useful in various cases.

Examples:

Keep tracks of calibration constants.
In the accelerator based long baseline experiments,
it will be very convenient to keep the beam timing
and the beam properties in the database.

But it is not necessary to use famous Relational Database.
(Oracle, PostgreSQL, mySQL etc.)

Most of the time, simple(?) data base system is enough.
ROOT, Berkeley DB etc…

Summary

- Even from the water Cherenkov type detector,
  total amount of data could be a few GB/sec.
  It is not impossible to handle this data with recent technology
  but the system should be designed carefully.
- The loss of the data is really critical in the analysis.
  To reduce the down time,
  the concept of HA system will be useful.
- The design of the network system is important.
  It is necessary to recognize the characteristic of the data flow
  in the online/DAQ system is different
  from the usual network traffic.
  Size of the packets might be a problem.
- Always keep in mind
  how to detect and investigate troubles.