A common real time framework for SuperKEKB and Hyper Suprime-Cam at Subaru telescope

---

CHEP'09, Mar 23-27 2009, Prague, Czech

Soohyung Lee, Korea University
Ryosuke Itoh, KEK
Nobuhiko Katayama, KEK
Sogo Mineo, University of Tokyo

Introduction

- SuperKEKB
  - In KEK, Tsukuba, Japan
  - Observation of CPV in B-meson decay in e^+e^- collision
  - Better understanding of CPV phenomena
  - Upgrade Belle experiment
    - Much larger data than Belle
      - Luminosity: ~$10^{36}cm^{-2}s^{-1}$
      - 50 billion BB-bar pair ($50ab^{-1}$)
      - ~250KB / event, 2GB / sec.

- HSC (Hyper Suprime-Cam)
  - Subaru telescope, Hawaii, USA
  - Search for Dark energy
  - Next generation CCD camera for Subaru telescope
  - Take the place of SC (Suprime-Cam) in 2011
  - 110 CCDs arranged in grid
  - 1 shot / 5 min. (including exposure time), 1.5GB / shot (880M pixels)
    - 1.5GB data for every 5 minutes
    - Data should be processed in 20 seconds to determine next exposure

Motivation

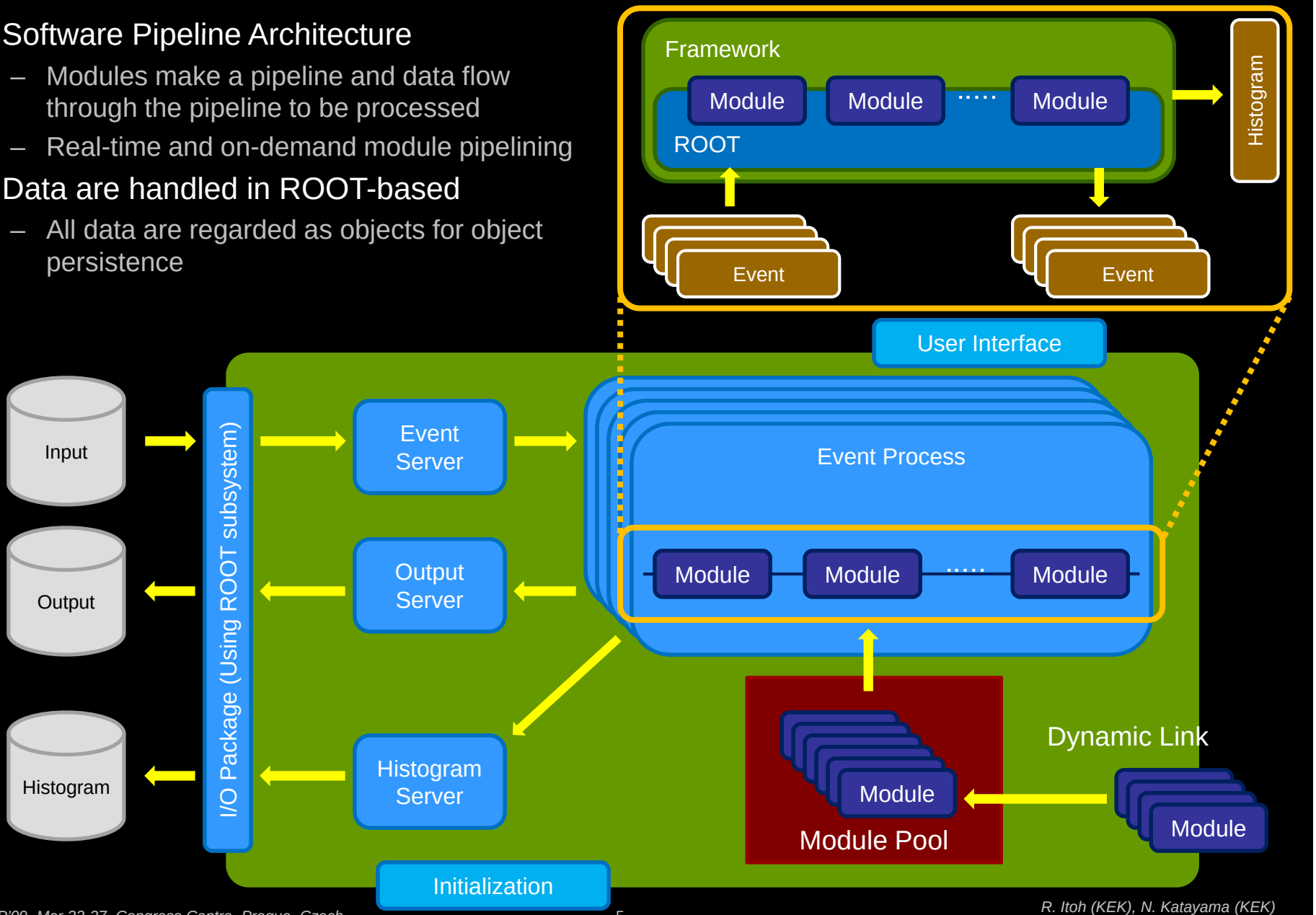
- Larger data to be processed for upgraded experiments
  - For SuperKEKB, ~100 times larger data than Belle experiment
  - For HSC, ~10 times larger data than SC
- Common interests in data processing between SuperKEKB and HSC
  - Real-time / On-demand processing parts
  - Module based data processing
- The limitation of current framework
  - Object persistence
  - Obsolete packages and environments
- New computing resources
  - Classic resources – SMP, network clusters
  - New resources – GRID, cloud computing and *et cetera*

Requirements

- Software Pipeline Architecture
  - Module pipelining for data processing
  - Real-time / On-demand module pipelining
- Object-oriented Framework
  - Provides convenient interface to the most common analysis tool – ROOT
  - Object persistence in data handling
  - Supports legacy interface
- Real-time and Parallel Processing
  - Huge data processing in real-time
  - Data / module parallel processing
  - Not only for the SMP but also network clusters and GRID
  - Module pipelining over network
  - Versatile and transparent parallel processing layer
  - Provide an interface for new computing resources
  - Dynamic optimization of resource allocation
- Integrated Environment
  - No boundary between DAQ and offline analysis
  - Integrated access method to ROOT files over GRID network

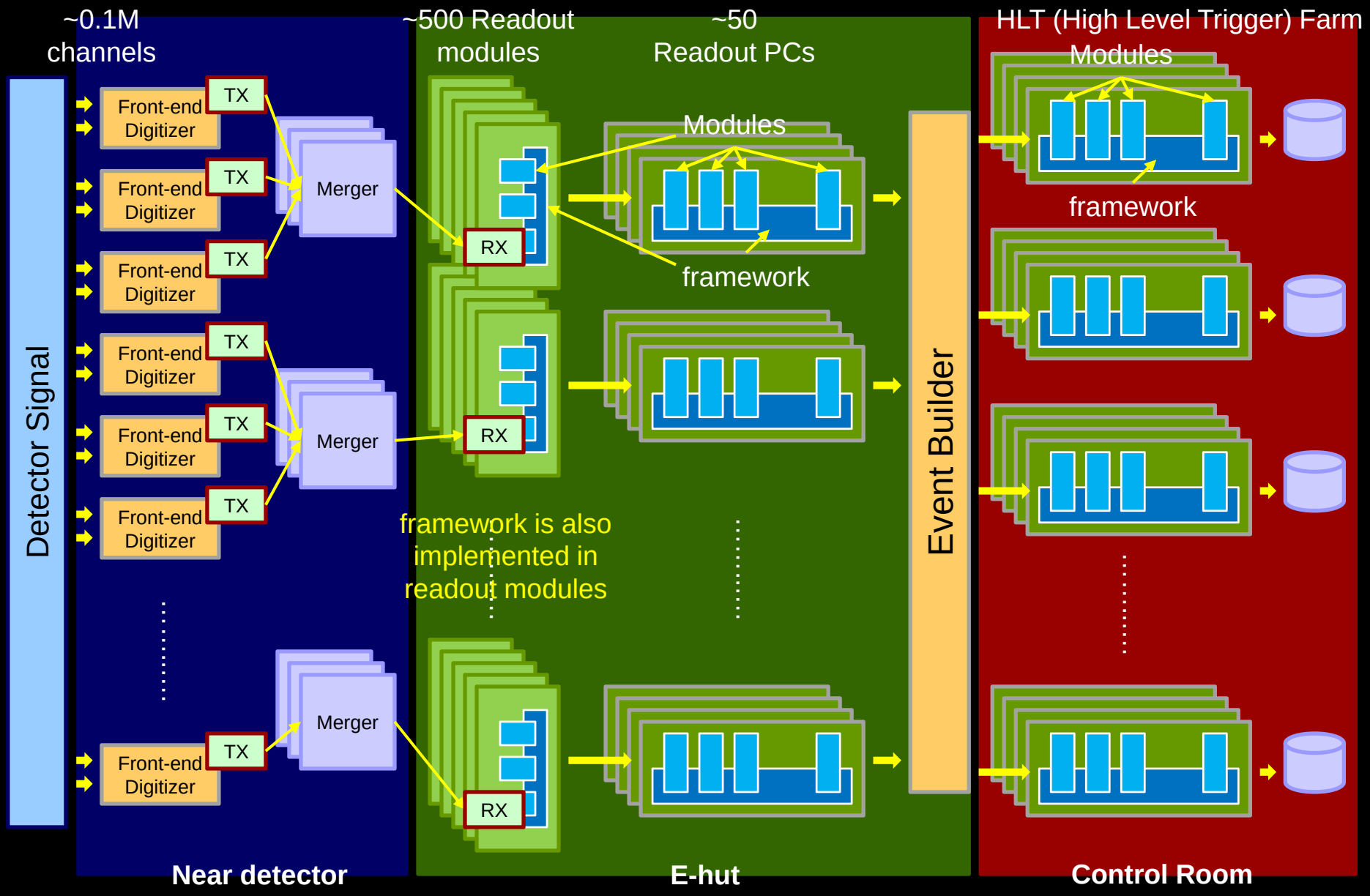
Architecture

- Software Pipeline Architecture
  - Modules make a pipeline and data flow through the pipeline to be processed
  - Real-time and on-demand module pipelining
- Data are handled in ROOT-based
  - All data are regarded as objects for object persistence



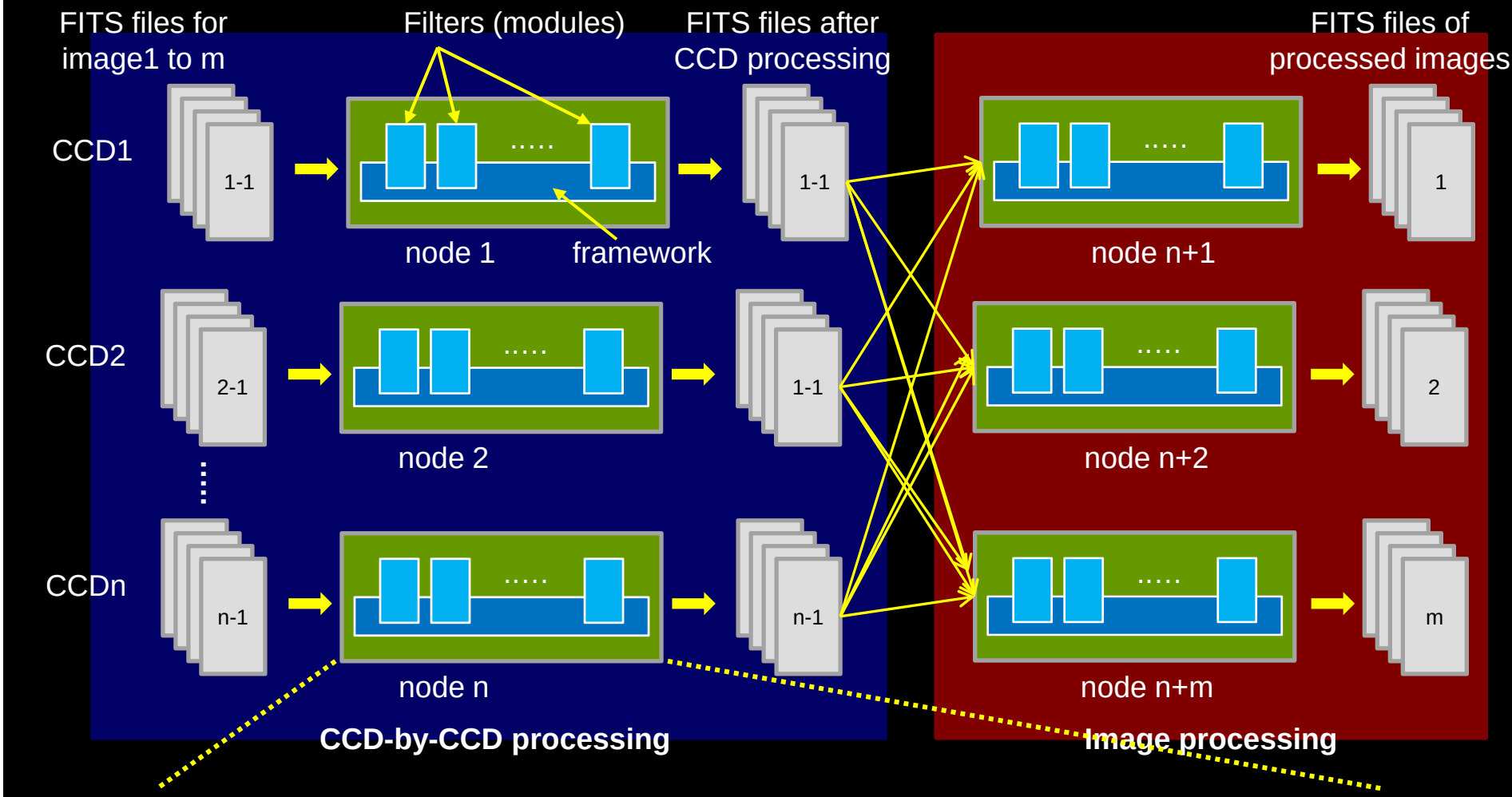
R. Itoh (KEK), N. Katayama (KEK)
S. Lee (Korea University), S. Mineo (University of Tokyo)

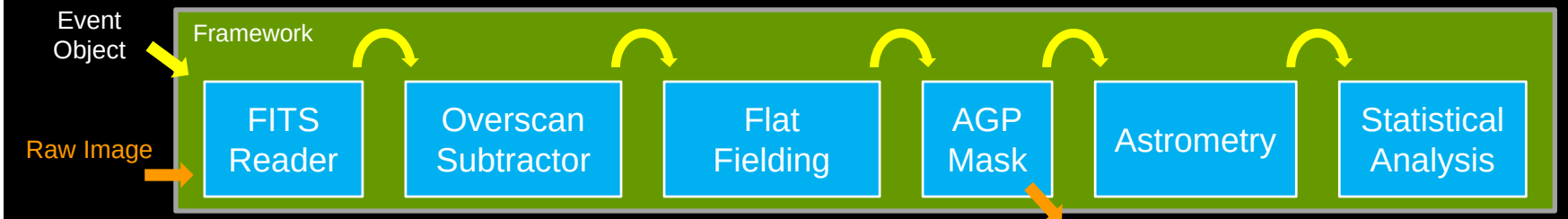
Real-time Pipelining in SuperKEKB



R. Itoh (KEK), N. Katayama (KEK)
S. Lee (Korea University), S. Mineo (University of Tokyo)

Real-time Pipelining in HSC





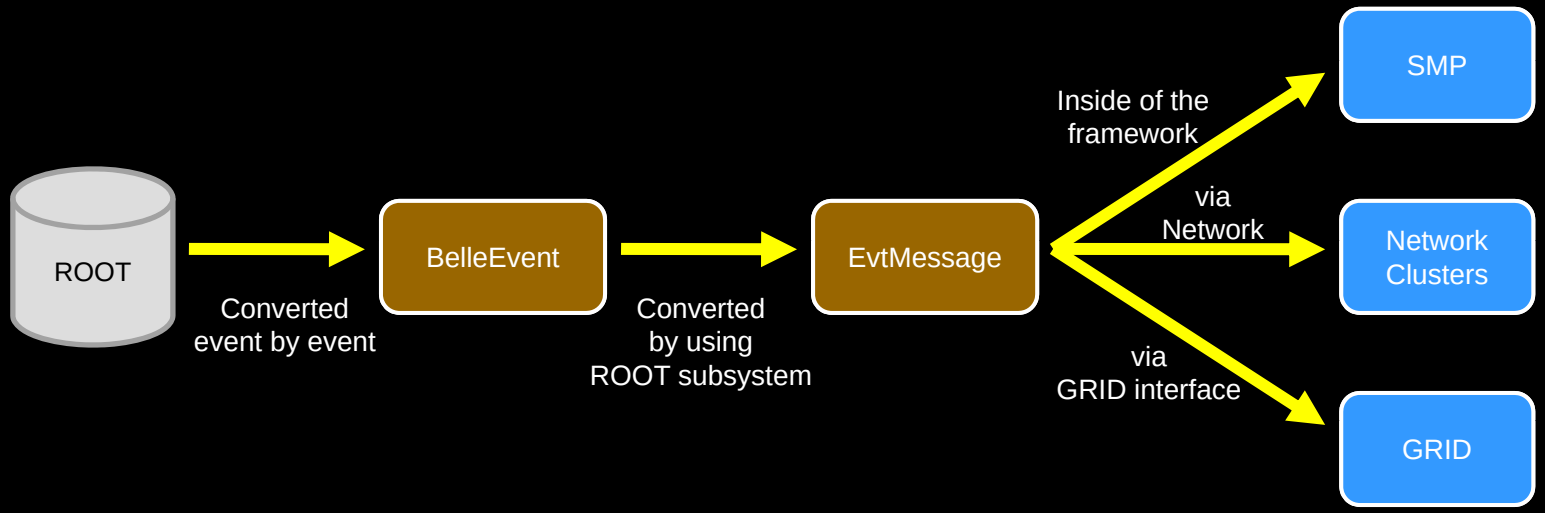
R. Itoh (KEK), N. Katayama (KEK)
S. Lee (Korea University), S. Mineo (University of Tokyo)

Object Persistence

- All data inside the framework are regarded as objects

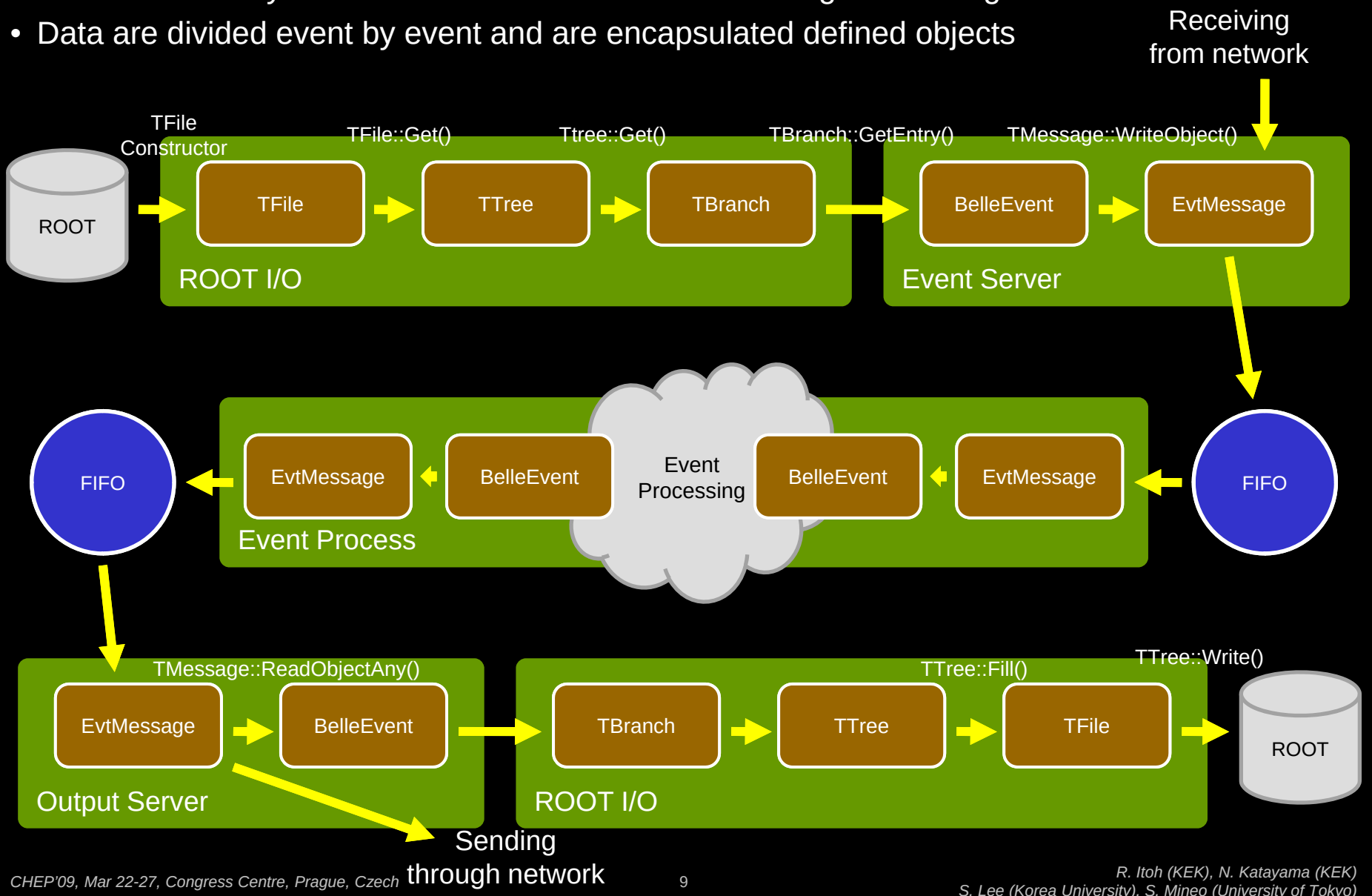
- BelleEvent Object
  - A basic unit of event data in user level
  - Input / Output ROOT files consist of BelleEvent object itself
  - Provides functions to get information from event data

- EvtMessage Object
  - A basic unit of event data for transferring inside of the framework
  - EvtMessage is derived from TMessage in ROOT
  - Event data are encapsulated as EvtMessage and transferred inside of SMP, through network clusters and GRID



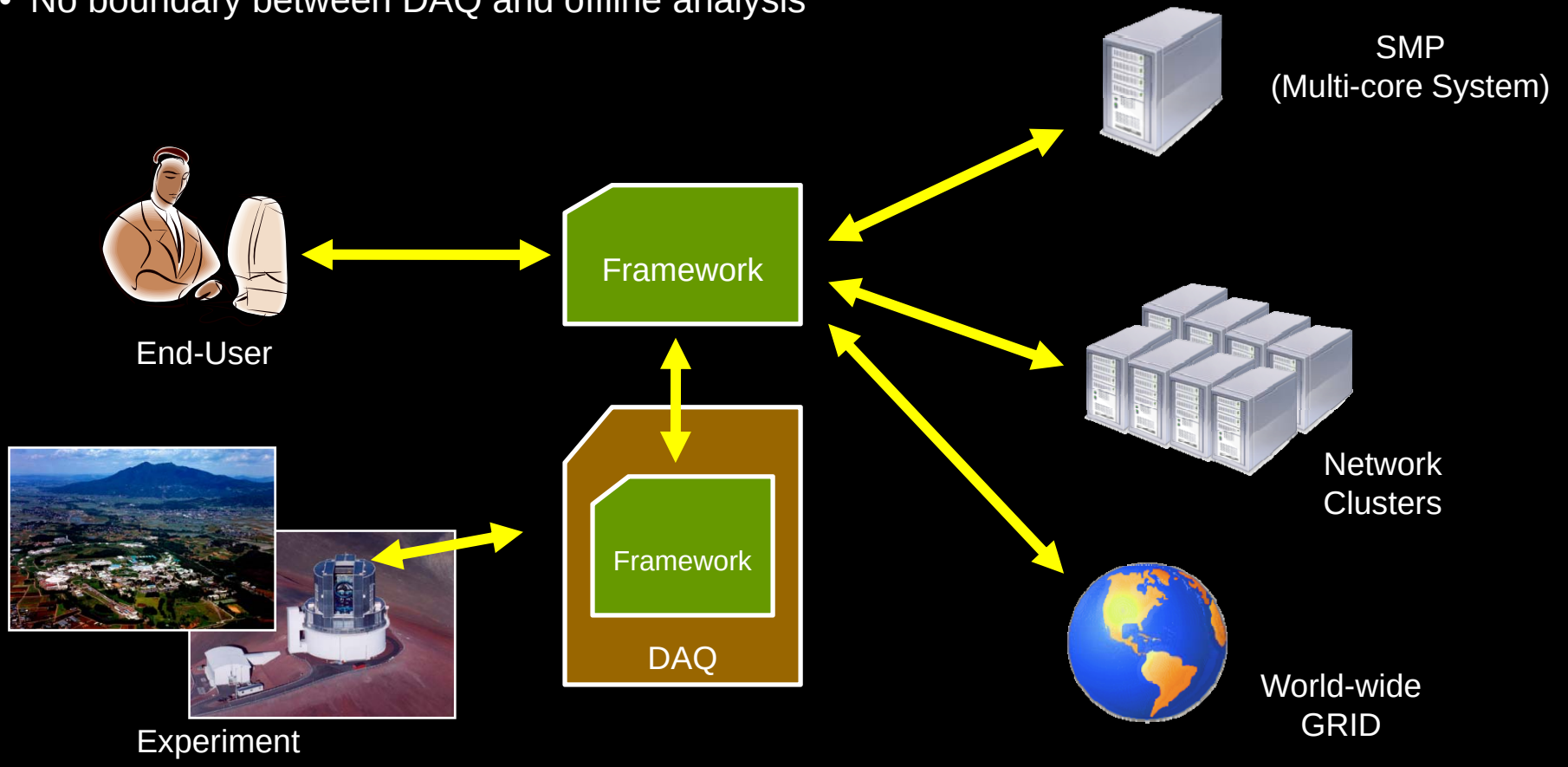
Object-oriented Data Flow in SMP

- The ROOT subsystem is used for ROOT I/O and building EvtMessage
- Data are divided event by event and are encapsulated defined objects

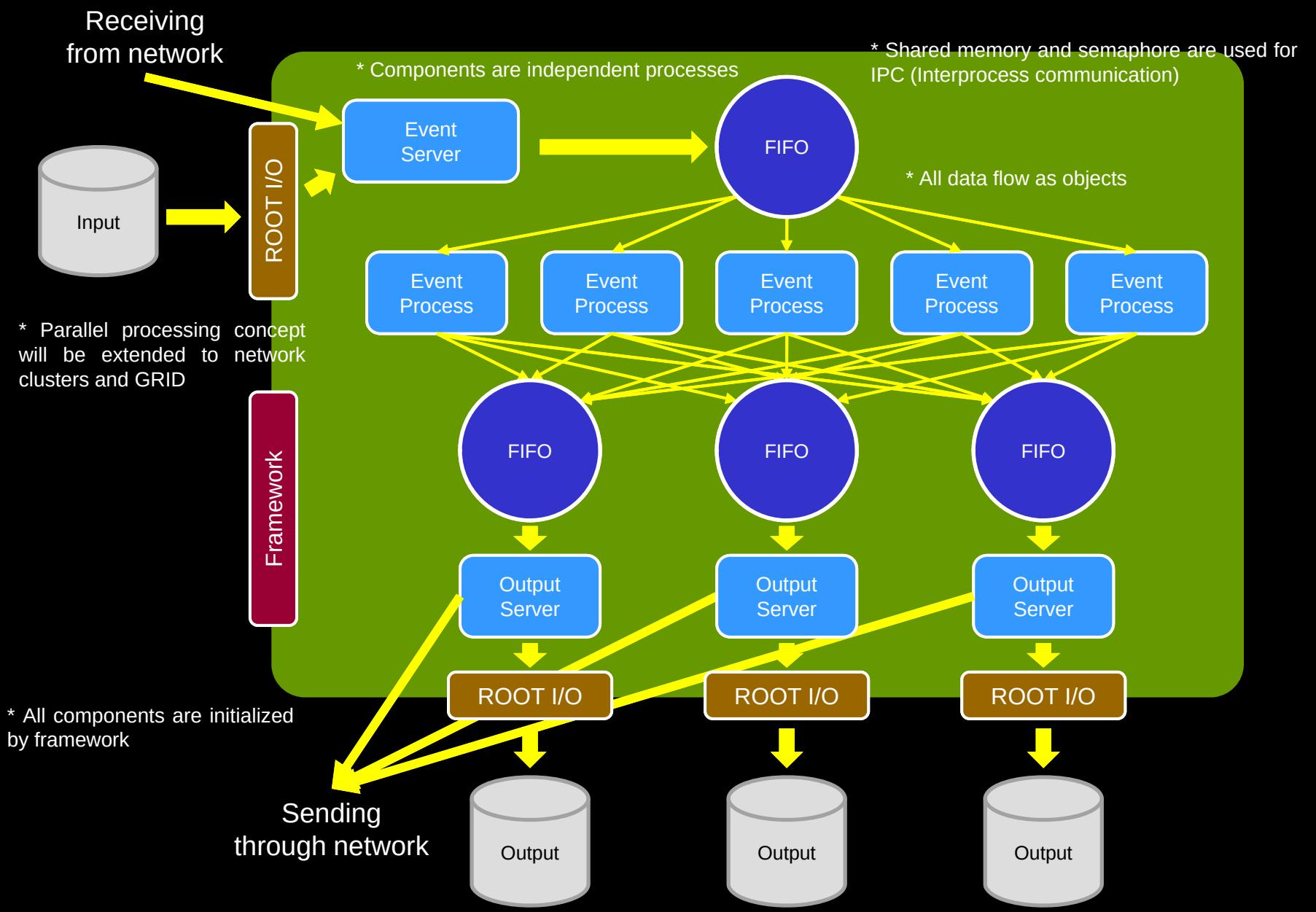


Parallel Processing Strategy

- New framework will provide consistent interface for parallel processing
- Dynamic resource selection among SMP, network clusters, and GRID by framework
- ROOT subsystem for networking (TSocket) will be used for network clusters
- GRID interface could be integrated into the new framework
- No boundary between DAQ and offline analysis

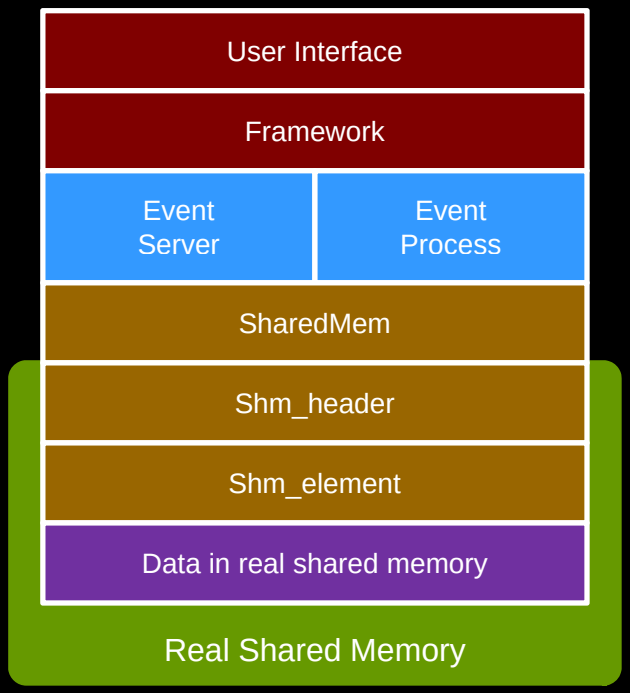


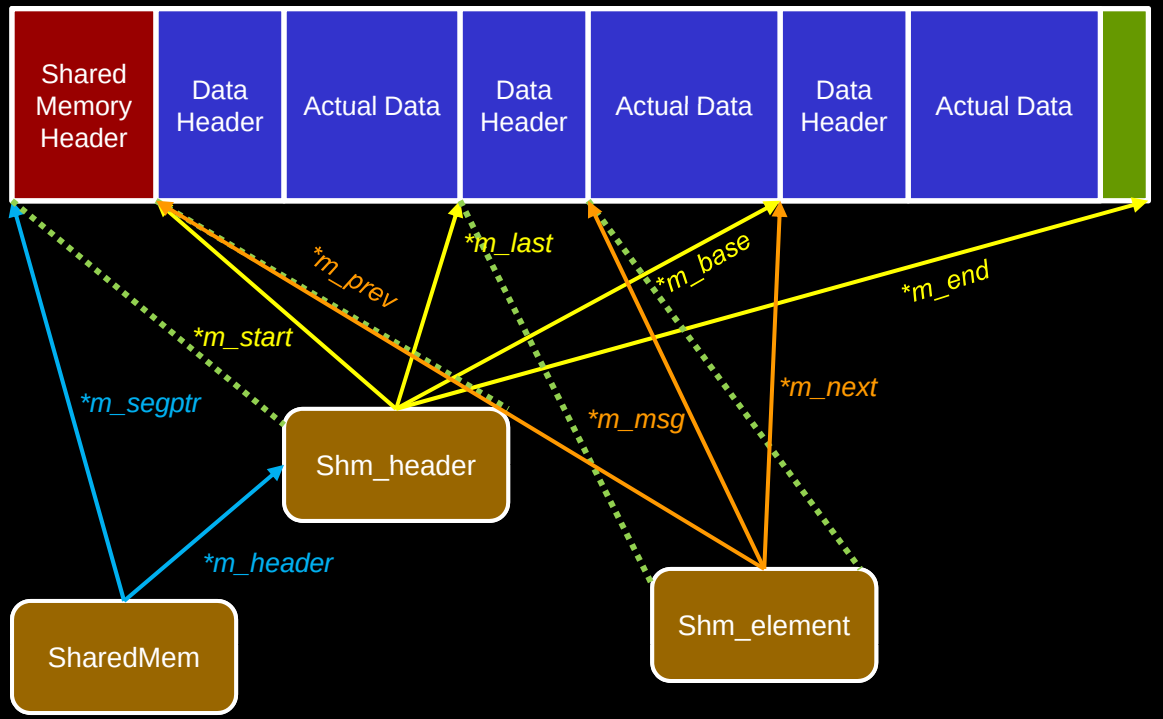
Parallel Processing Scheme in SMP



Shared Memory as FIFO (First-In First-Out)

- Logical Structure of shared memory as FIFO
  - Layers make a hierarchy
- Each components has pointers to handle shared memory scheme
- This logical structure provides behavior of FIFO
  - Earlier input data is sent first
  - Similar to ring buffer



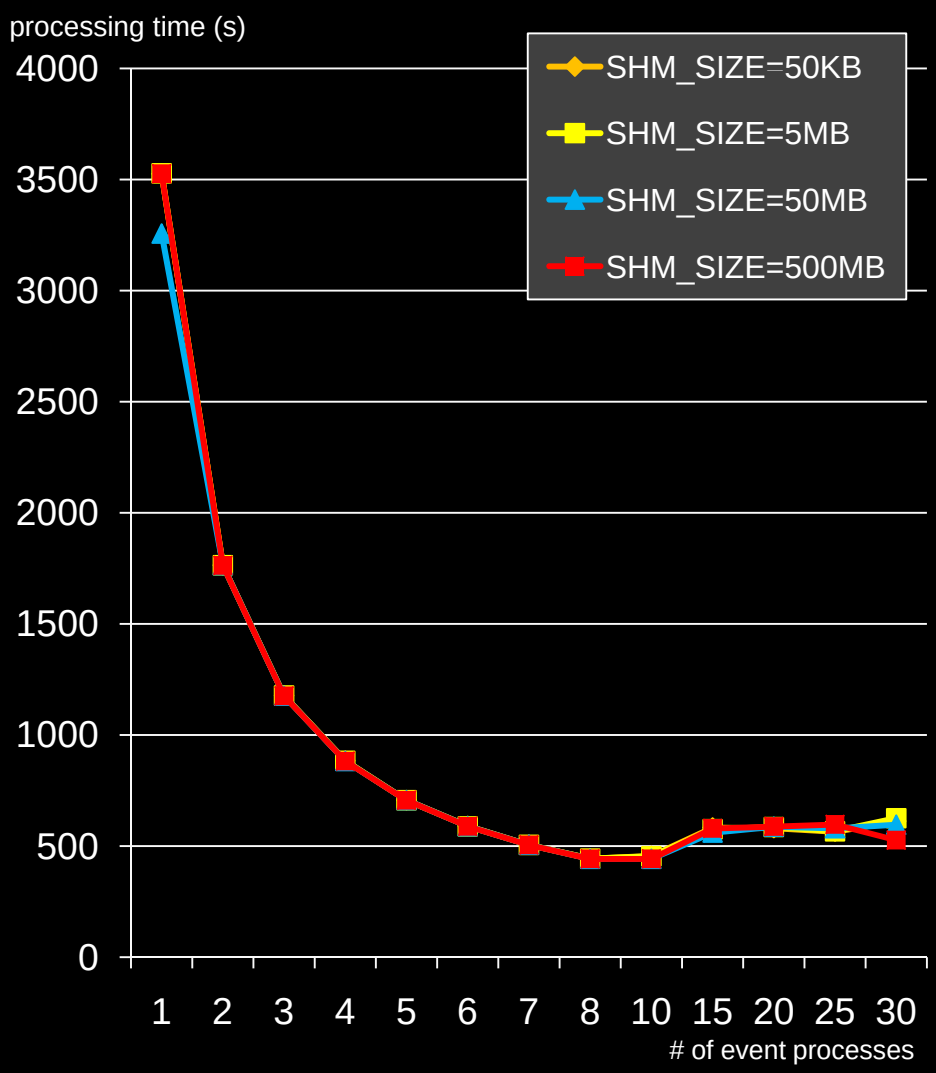


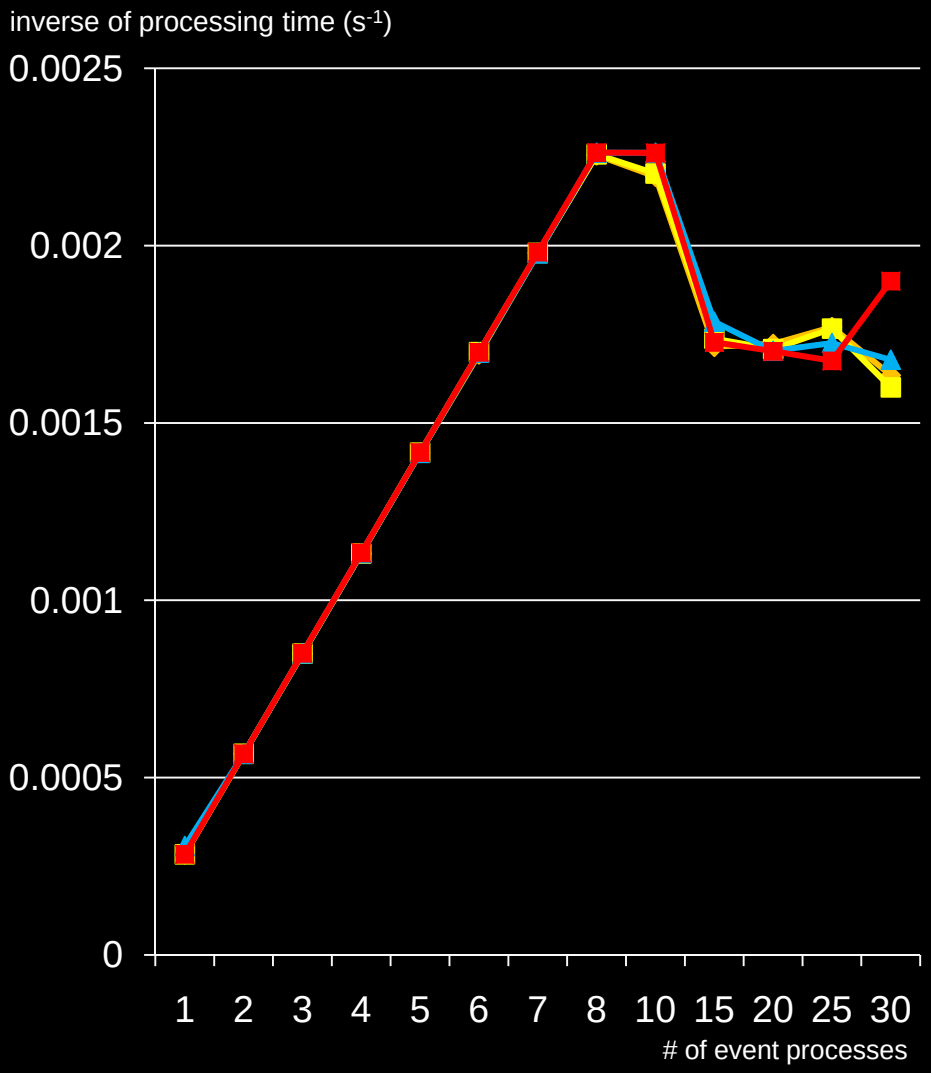
Current Development Status

- Fundamental components of the framework are implemented
  - Interfaces to handle modules, I/O
  - Many parts of codes are reused from the old framework of Belle experiment
- Object-oriented data flow scheme is designed and implemented
  - By combining ROOT subsystem, object persistence in the framework is implemented
  - Objects are designed for not only SMP but also distributed computing systems
- Parallel data processing
  - Designed and implemented for SMP first
  - Customized shared memory is designed and implemented
  - Integration parallel processing part (including shared memory) to the framework
  - Functionalities are tested in various environments
  - Performance behaviors for various # of CPUs are observed by testing
- Parallel module pipelining
  - Integration HSC modules to the framework
  - Real-time image processing is tested

Performance Test – Time Consumption of Modules

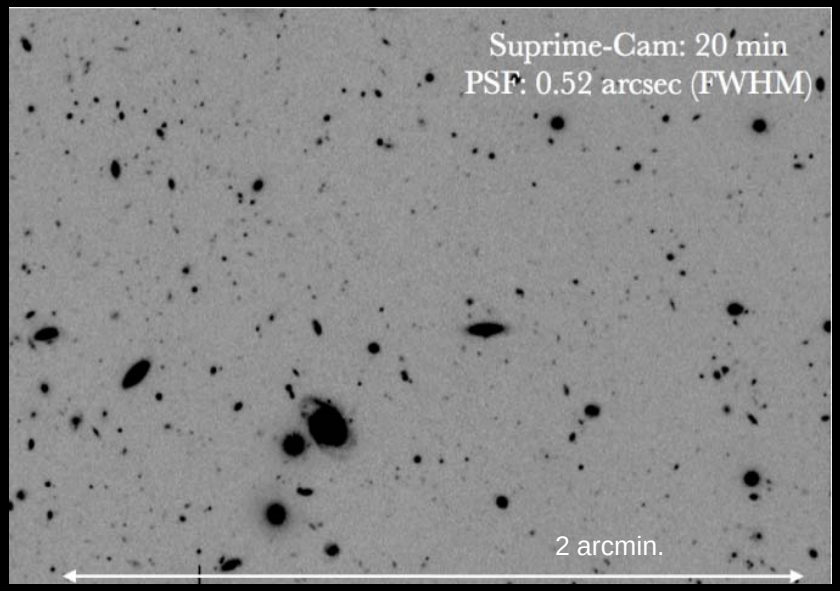
- 8-core system (Xeon E5405 (2Ghz, Quad-core) * 2, 8GB RAM)
- # of events to be processed: 100,000 events (~300KB per events)

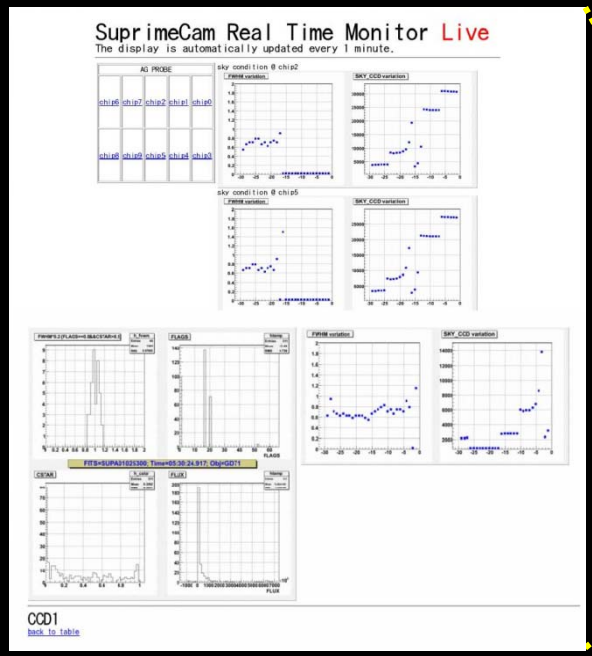




Functionality Test in SC (Suprime-Cam)

- The framework is combined with HSC modules and the functionalities are tested





WEB

Analysis/visualization using ROOT

Suprime Cam DAQ

FITS files for one CCD

Output FITS

N-tuple

Framework

FITS reader | Overscan Subtractor | GetPar | Flat Fielding | AGP Mask | Sextractor | Stat

Histogram Manager

ROOT Object (on memory)

Summary

- Upgraded experiments require more powerful framework
  - SuperKEKB and HSC
- The framework is based on software pipeline architecture
  - Real-time / On-demand pipelining
- Key functionalities
  - Object persistence
    - ROOT-based data handling
    - Convenience for both of end-users and developers
  - Parallel processing
    - Parallel data processing and module pipelining
    - Provides an integrated interface to local computing resources – SMP and network clusters
    - A bridgehead for using world-wide computing resources
- The framework is being developed
  - Parallel data processing in SMP is developed and tested
    - Implementation in network clusters by this fall
    - Implementation for GRID by early of next year
  - Parallel module pipelining over network is developed and tested