PROCEEDINGS OF SCIENCE

Distributed parallel processing analysis framework for Belle II and Hyper Suprime-Cam

Sogo Mineo[*a], **Ryosuke Itoh**[b], **Nobuhiko Katayama**[b], **Soohyung Lee**[c]
[a]*Department of Physics, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan*
[b]*IPNS, KEK, 1-1 Oho, Tsukuba, Ibaraki, Japan*
[c]*Department of Physics, Korea University, 1 Anam-dong, Seungbuk-gu, Seoul, Korea*
E-mail: mineo@hep.phys.s.u-tokyo.ac.jp, ryosuke.itoh@kek.jp, nobu.katayama@kek.jp, shlee@hep.korea.ac.kr

We report the development of the analysis framework named ROOBASF. ROOBASF is originally the framework for the future Belle II analyses, but is also planned to serve as the framework for HSC analyses. We have enhanced ROOBASF for its sake to have a capability of MPI distributed-memory parallel processing. We have implemented a Python interface on ROOBASF, offering users an efficient way of developing their analysis modules. We ported to ROOBASF an analysis pipeline for primary treatments of astronomical images. The pipeline was successfully parallelized with ROOBASF.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research
February 22-27, 2010
Jaipur, India

[*]Speaker.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PoS(ACAT2010)026

1. Introduction

The Subaru Telescope with its primary mirror of 8.2 meters in diameter is located at the summit of Mauna Kea, Hawaii. The telescope can mount a camera on its prime focus offering a wide field of view (FOV). The next-generation camera to be mounted thereon is named Hyper Suprime-Cam, or HSC [1], the first light due in the autumn of 2011. The HSC has an FOV of 1.5 degrees in diameter, $\sim$7 times larger than the current camera, Suprime-Cam (SC) [2]. With that large FOV, we are planning to carry on a sky survey of 2000 deg^2 to investigate dark energy. On the focal plane of HSC are 116 CCD sensors laid out in a disk shape, the resolution of each CCD being 4k $\times$ 2k. An exposure of the HSC produces data of $\sim$1G pixels, or $\sim$2G bytes, urging the necessity of developing efficient analysis system.

On the other hand, KEK is planning SuperKEKB and the Belle II experiment [3]. SuperKEKB being the upgrade of the KEKB collider, and Belle II of the Belle detector [4][5] , the two aim for better understanding of CP violation in B meson decays with more luminous asymmetric e^+e^- collisions. The planned luminosity being $10^{36}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, the data rate will be 6GB per second right after the L1 trigger, and 600MB per second after online reconstruction. To perform the on-line analyses, and off-line ones as well, a framework named ROOBASF [6] is developed at KEK, based on the Belle AnalysiS Framework (BASF) [7], which have been used in the Belle experiment.

We plan to use this ROOBASF as the framework of on-line and off-line analyses in HSC. The property of HSC data is, however, different from that of Belle II. In Belle II, the generated events are very frequent but each is rather small, and the events are independent of each other. ROOBASF accordingly has an event-parallelizing function during execution, where many events are processed simply in parallel.

Meanwhile, the HSC produces at once $\sim$2GB of data, which consist of $\sim$100 images from the respective CCDs. We further superimpose many shots of duplicative sky regions to achieve higher S/N. The images analyzed here depend on each other, unlike Belle II events, because they reflect respective parts of the same region of the sky. We thus need to analyze these many images in parallel but not independently, often having the analyzing processes communicate with each other. We describe the enhancement of ROOBASF to perform algorithm- and program-parallel processing necessary for the HSC analyses.

2. ROOBASF

The Belle AnalysiS Framework (BASF) is the framework for the Belle experiment having successfully been used for 10 years. BASF has been involved in nearly all of the experiment, including data acquisition, simulation, and users' analyses. BASF has a software pipeline architecture (Figure 1), where pipeline modules are linked dynamically, enabling flexible modular structure of analysis paths.

We are revising BASF to a new framework named ROOBASF for Belle II. Among current high energy experiments, the ROOT library [8] is the most popular analysis tool. We have embedded the ROOT in ROOBASF and achieved object-oriented data flow and persistence.

As we describe in detail below, we have further introduced the Message Passing Interface (MPI) to ROOBASF. ROOBASF can then make data-parallel analysis paths with inter-process

PoS(ACAT2010)026

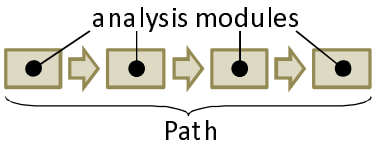

Figure 1: BASF software pipeline

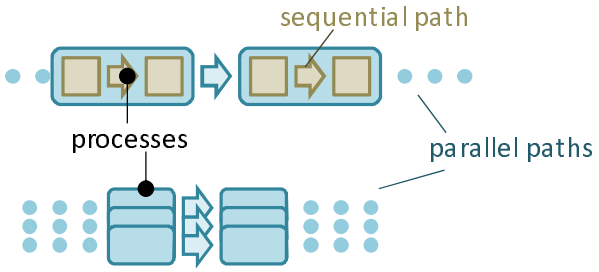

Figure 2: ROOBASF parallel paths

communication, and split paths along the way to make program-parallel paths (Figure 2). In addition, we implemented a Python interface on ROOBASF, which makes module development efficient.

Parallelization To parallelize analysis paths, we use the MPI, which defines the de-facto standard communication API in distributed parallel computing. Not only the framework itself, but also analysis modules can use MPI to perform algorithm-parallel programs. Processes in parallel running a stage of a pipeline are grouped, and the group is by ROOBASF assigned with an MPI communicator, with which the program of the stage can execute parallel algorithms without interference with the other processes running different analysis programs.

Analysis path BASF has a concept of a *path* as shown in Figure 1. A path is a sequence of analysis modules, all of which are sequentially executed in a process. Paths are optionally connected to others with branching conditions, and these connected paths are also sequentially executed. In ROOBASF, as illustrated in Figure 2, we have implemented another layer of path named *parallel path*, and the traditional path is renamed *sequential path*. A parallel path is analogous to sequential path, but is a sequence of processes, instead of modules, with optional conditional branches. Each of the processes in a parallel path runs a sequential path, resulting in program-parallelization. Multiple copies of the parallel path can further be run at a time, thereby achieving data-parallelization. These two types of parallelization are what HSC analyses have required.

Python utilization ROOBASF also employs the Python language, in which path configuration is described. The python interface is implemented with boost.python in the boost C++ library. Boost.python not only enables Python to call ROOBASF in native codes, but also allows ROOBASF to call Python analysis scripts, which permits us also to describe analysis modules on the Python script in line. Analysis modules can thereby be developed easily in Python. If the analyses are costly, we can re-write the modules in C++. Hence the framework allows for the efficient development of analysis modules.

3. Test pipeline for HSC

We built on ROOBASF an analysis pipeline for astronomical images, by porting one from a prototype of the on-line monitoring system being developed for HSC. We data-parallelized the pipeline, each process of which was given an image to analyze. We show the pipeline in Figure 3.

PoS(ACAT2010)026

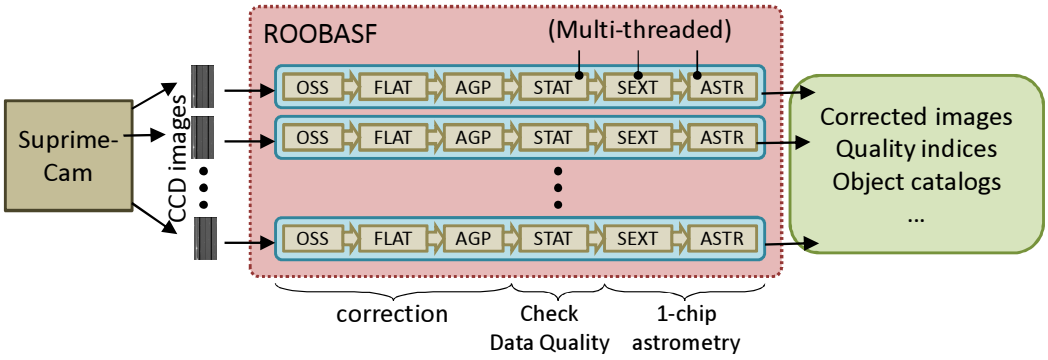

Figure 3: Test pipeline

We intend that the pipeline in future analyze simultaneously ∼100 CCD images of an exposure, but, in the test below, 9 images at most were concurrently processed.

The process, given input images, performs pedestal and gain correction of the image. The pipeline continues to check data quality for the on-line monitoring, and then performs 1-chip astrometry. In the astrometry, we needed to access a catalog of celestial objects on the Internet. The astrometry processes, using MPI communication, delegate the Internet access to one of themselves, so as to suppress a denial of service. In addition, there exist some tiny modules written in Python, such as an error detector, a time watch, and so forth. The last three modules are multi-threaded so that the execution is made as fast as possible.

We ran the pipeline in the following environment and got the result as we describe thereafter.

Test environment We used three PCs, each of which had a quad-core CPU. We linked them through the Gigabit Ethernet. Programs and libraries were shared among the three using NFS, while input and output images were stored on local disks. As seen in Figure 4, we tested our framework with the number of processes equal to 1, 3, 6, and 9. Since we had only three PCs, we still had to assign two or three processes on each PC when the number of processes was six or nine.

Result The execution time to process an image was measured and is shown in Figure 4. The elapsed time per image is reciprocally plotted against various numbers of processes (the red line). On the right axis we show the corresponding speedup with respect to the sequential processing. The ideal speedup is represented by the straight green line.

The measured speedup could not keep up with the ideal line because of the shortage of CPU cores. When we used six or nine processes concurrently, we had to assign two or three processes on each PC, while the pipeline contained multi-threaded modules. Each process tried to use all the CPU cores, and the speedup did not become linear.

Nonetheless, we were able to process images 4.5 times more quickly with 3 PCs when 9 processes were used.

4. Summary

We parallelized the analysis framework, ROOBASF, for HSC as well as Belle II. The frame-

PoS(ACAT2010)026

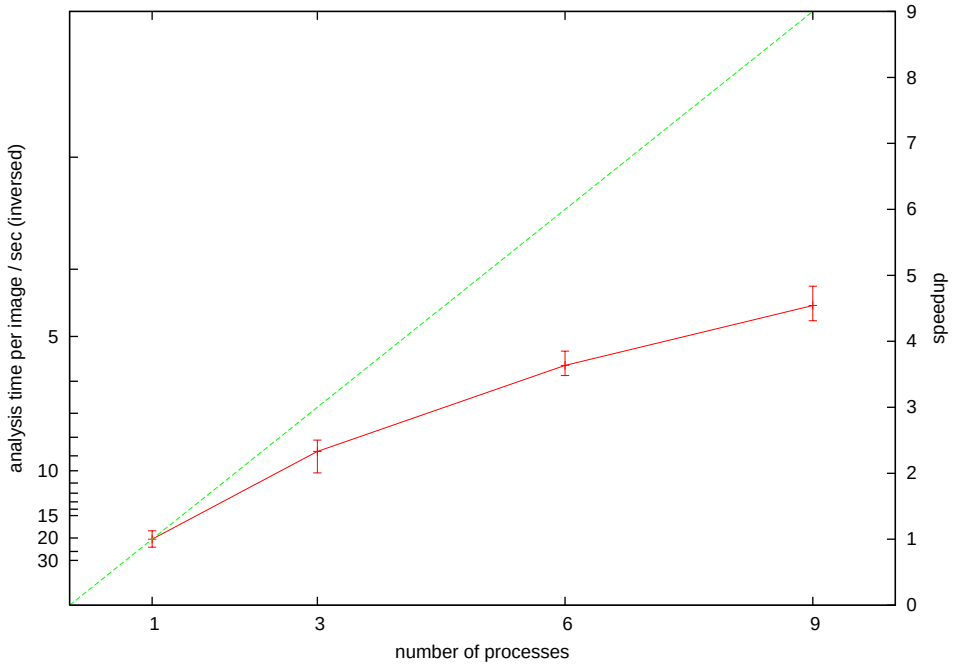

Figure 4: Speedup of the test pipeline according to the number of processes

work uses MPI for distributed-memory parallel processing, and we implemented the Python interface on the framework. We then successfully built the test parallel pipeline for astronomical images, though the speedup was not good enough because of CPU shortage. The framework is still under development for quick module development and quick analysis.

References

[1] S. Miyazaki et al., *HyperSuprime: Project Overview*, *Proc. of SPIE* **6269** 62690B

[2] S. Miyazaki et al., *Subaru Prime Focus Camera—Suprime-Cam*, *Publ. Astron. Soc. Jpn* **54** 833–853

[3] K. Abe et al., *Letter of Intent for KEK Super B Factory*, http://belle.kek.jp/superb/loi/

[4] A. Abashian et al., *The Belle detector*, *Nucl. Instr. and Meth.* **A479** 117-232

[5] I. Adachi et al., *Belle computing system*, *Nucl. Instr. and Meth.* **A534** 53-58

[6] S. Lee, *A common real time framework for SuperKEKB and Hyper Suprime-Cam at Subaru telescope*, *International Conference on Computing in High Energy and Nuclear Physics (CHEP 2009)* to be published

[7] R. Itoh, *BASF - BELLE AnalysiS Framework, Talk given at Computing in High-energy Physics (CHEP 97), Berlin, Germany, 7-11 Apr 1997*, http://www.slac.stanford.edu/spires/find/hep/www?irn=3592502

[8] F. Rademakers and R. Brun, *ROOT: An Object-Oriented Data Analysis Framework*, *Linux Journal* **51**

PoS(ACAT2010)026