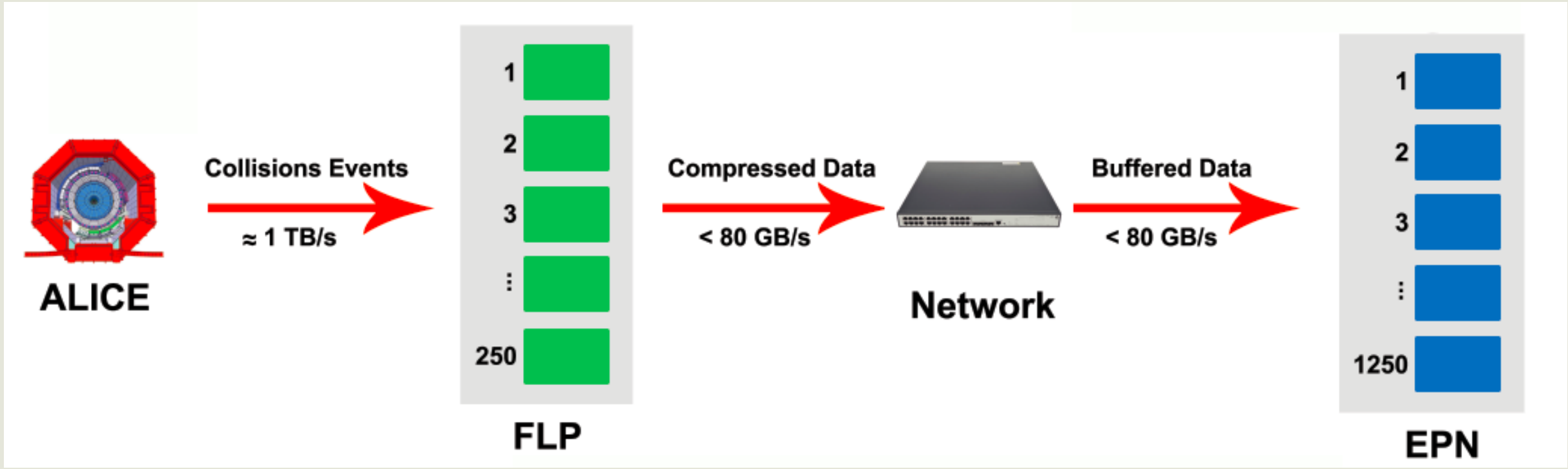
The O^2 Developments in KMUTT

Tiranee Achalakul
King Mongkut's University of Technology Thonburi

ALICE O^2

- ALICE detector will be upgraded in 2018
  - Handle 50 kHz Pb-Pb collisions
  - Higher data throughput from the detector (1 TB/s)
- Data is processed both **O**nline and **O**ffline (O^2)
- Currently, in search of a suitable computing platform and a good scheduling framework.

Computing Process



Computing Nodes

- FLP (250 nodes)
  - Perform reduction on data fragments
  - Decrease the size by 5x -10x
- EPN (1250 nodes)
  - Perform Calibration, Event Reconstruction, Data Compression
  - Decrease to size by 2x

KMUTT Involvement

- A research team from the computer engineering department at KMUTT has planned a collaborative research framework under the ALICE O^2 project.

- Goal: select the suitable computing platforms by optimizing the size and the cost of the online farm.

- The collaboration began earlier in 2014
  - Two graduate students have started working on the project (Mr. Boonyarit Changaival, Miss Sarunya Pumma)
  - CERN Mentor has been assigned (Mr. Sylvian Chapeland)

Two of our sub-projects under ALICE O^2

- Assessing the performance of different types of hardware and programming models.
  - Pixel Cluster Finding on GPU
- A dynamic scheduling framework for online data processing
  - Select computing nodes inside the Event Processing Nodes (EPNs) cluster
  - Delegate a large number of jobs from the First Level Processors (FLPs)
- Both research works are still in an early stage.

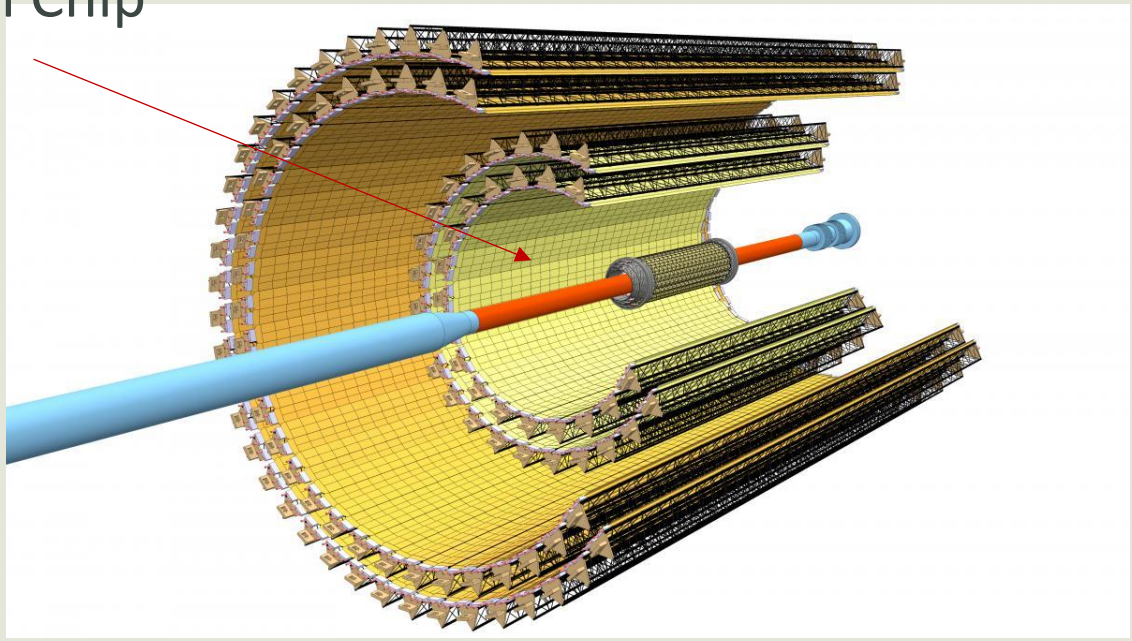
Pixel Cluster Finder on GPU

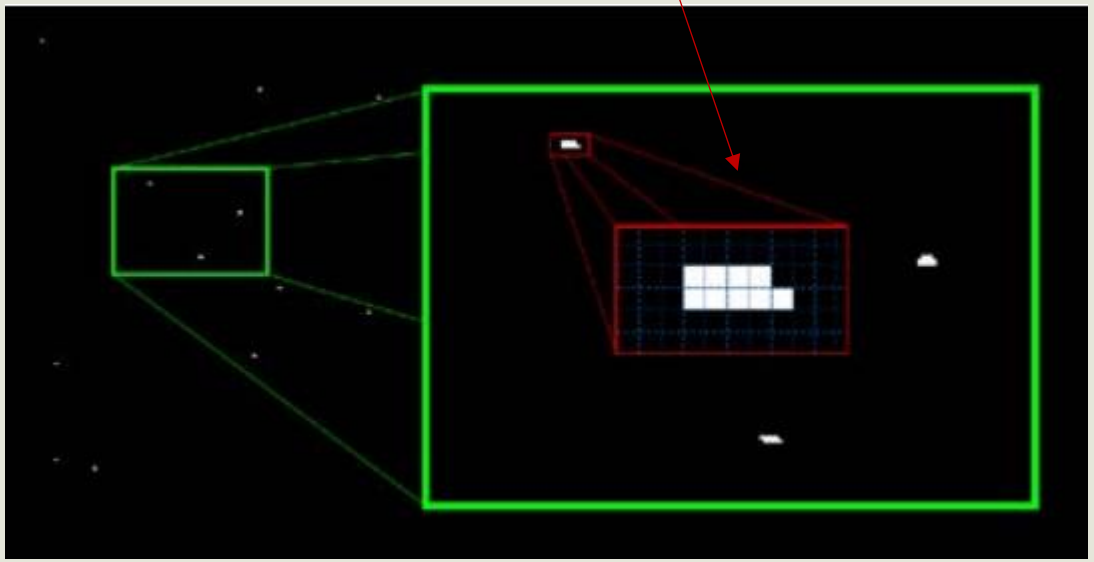
Pixel Cluster Finder

- Purpose
  - Identify groups of adjacent pixel hits
    - 360 hits per chip (average)
    - 430 chips in the detector
  - Compute their center of gravity
- Input
  - A list of 2D hits coordinates in pixel row and column
- Output
  - A list of clustere 2D coordinates in millimeters from chip center
    - Average of 60 clusters per chip

ITS Detector & Pixel Cluster

Pixel Chip





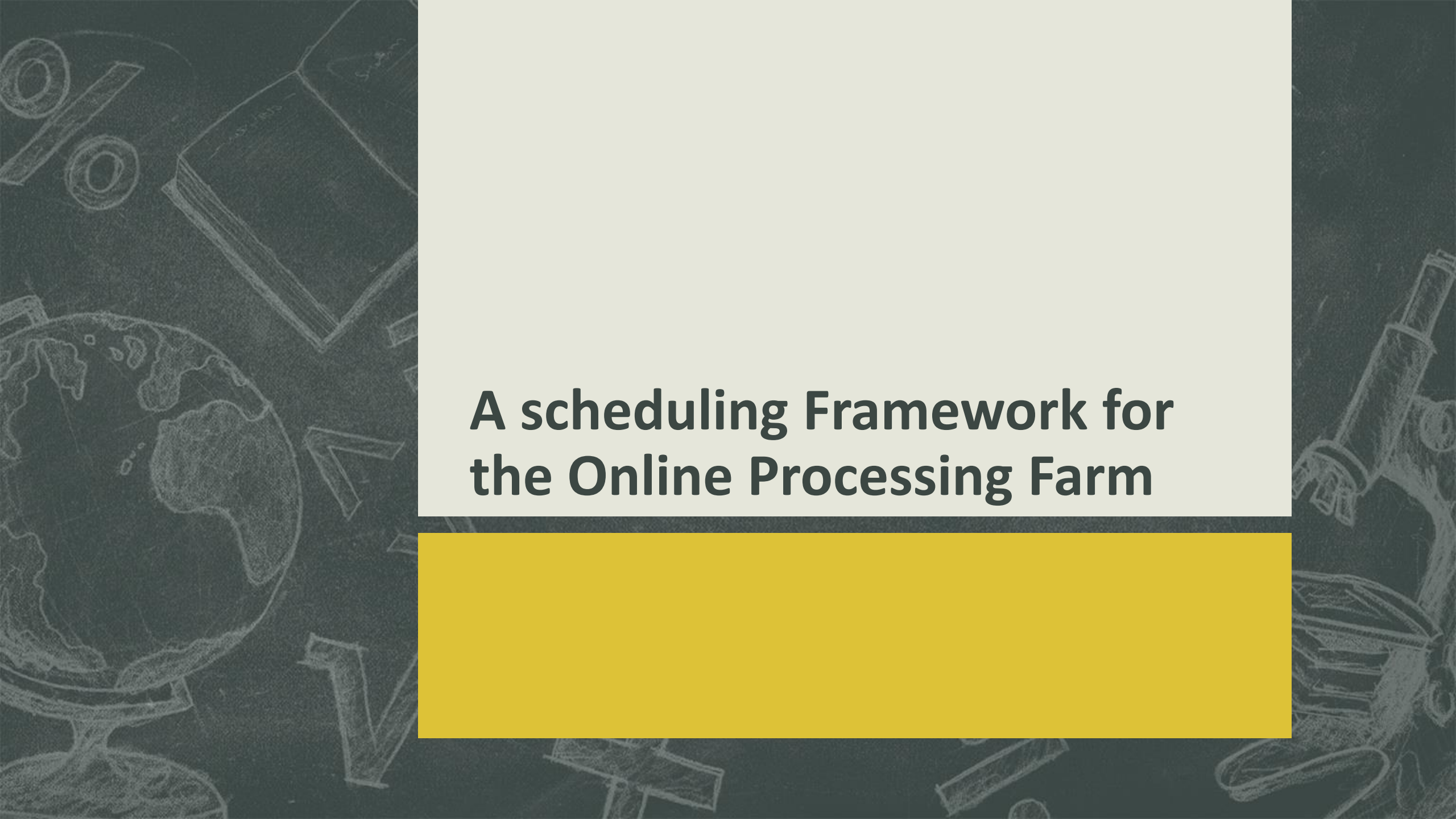
5 steps in the Pixel Cluster Finder algorithm
Pre-grouping, Assigning Cluster, Joining Cluster, Cluster Summation, Calculating Center

Pixel Cluster Finder on Multiple Platforms

- Exploit the multicore-architecture
- Attempt on the Xeon Phi architecture
- Initial works on the Nvidia GPUs (By KMUTT)
  - Redesign the algorithm to fit the GPU's memory hierarchy
  - Evaluate the performance on a few GPU systems
  - Optimize the performance on the GPU

Performance

- Initially tested on GTX 780 with the Nvidia Insight profiler
  - High occupancy in most kernel (> 90%)
  - Very low on serialized portion
- Process around 30 events in 1 second
- Further Tuning
  - Modify the data structure to better suit the GPU architecture
  - Use CUDA specific commands for efficient memory allocations
  - Performance evaluation on the Tesla K20 or K40 system



A scheduling Framework for the Online Processing Farm

A Scheduling Framework

- A scheduling framework for distributing tasks from FLPs to EPNs
- Optimize 2 objectives
  - Makespan
  - Energy Consumption
- The work had been presented in the poster session at PASC 14 conference in Zurich during 2-3 June, 2014.

Challenges

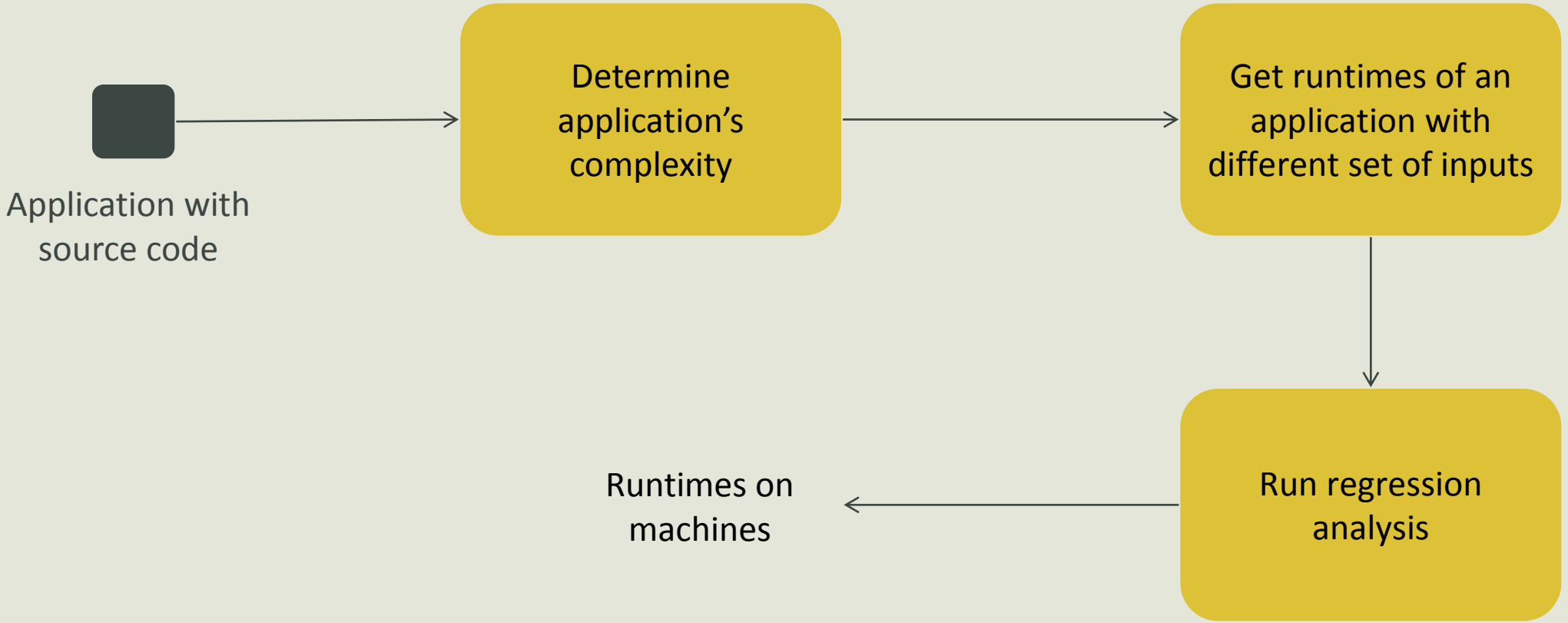
- Job runtime estimation
  - Must be accurate
- Fast and efficient scheduler is needed
  - Delays cause processing bottleneck and increase the buffer storage needed on the FLP cluster.

Scheduler Functions

- Runtime estimation
  - Complexity analysis
  - Regression model
- Scheduling approach
  - Round-Robin
  - Meta-heuristic for multiple objectives optimization

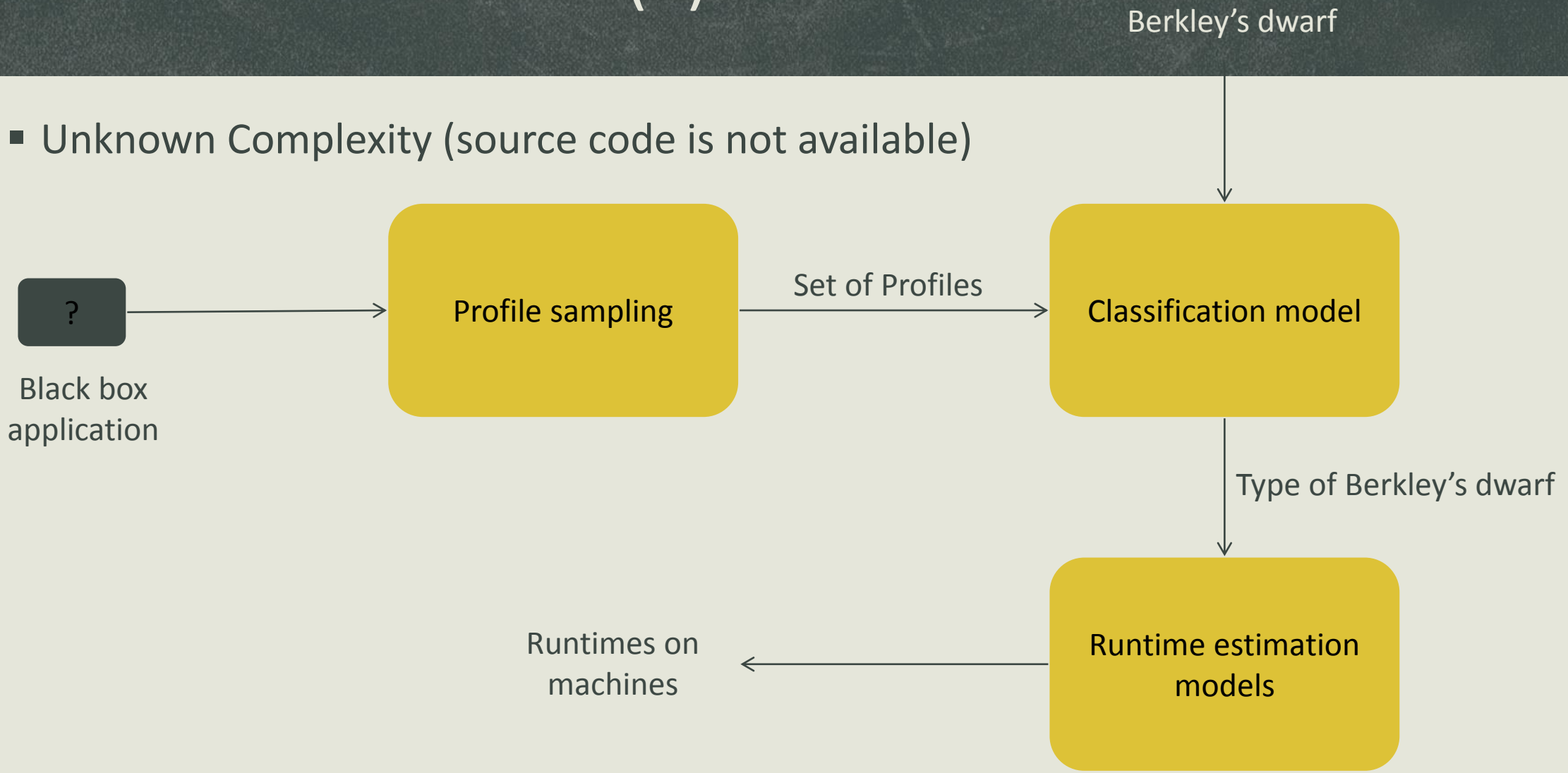
Runtime Estimation (1)

- Known Complexity (source code is available)



Runtime estimation (2)

- Unknown Complexity (source code is not available)



Runtime Prediction: Unknown profile

Profile Collecting Process

- Tools
  - MICA - Microarchitecture-Independent Characterization of Applications
    - Machine architecture independent
    - Compiler dependent
    - Characterize the profile of process using 8 metrics
  - Perf – Profiler tool for Linux 2.6+
    - Machine and compiler dependent
    - Collect 4 software events

Runtime Prediction: Unknown profile

Profile Collecting Process

- List of profiles
  - Probability of a register dependence distance <= 16
  - Branch predictability of per-address, global history table (PAg) prediction- by-partial-matching (PPM) predictor
  - Percentage of multiply instructions
  - Data stream working-set size at 32-byte block level
  - Probability of a local load stride = 0
  - Probability of a global load stride <= 8
  - Probability of a local store stride <= 8
  - Probability of a local store stride <= 4,096
  - CPU clock
  - Task clock
  - Page faults
  - Context switches

MICA

Perf

Runtime Prediction: Unknown profile

Classification Model

- Classify the unknown application into a type of Berkley's dwarf
- 13 Berkley's dwarfs – represent characteristics of the scientific applications
  (i.e. dense linear algebra, sparse linear algebra, and n-body methods)
- Train the model
  - Run the benchmark of the Berkley's dwarfs on the 1-core computer with Ubuntu operating system
    - NAS Parallel Benchmarks (from NASA)
    - Rodinia (from Virginia Tech)
    - TORCH (from University of California)

    } Compiles with *gcc, g++*
  - Collect data records (use only MICA metrics)

Runtime Prediction: Unknown profile

Classification Model

- Train the model
  - Use C4.5 algorithm to build the decision tree
- Validate the model
  - Using 10-fold cross validation
  - The accuracy is above 96.89%

Runtime Prediction: Unknown profile

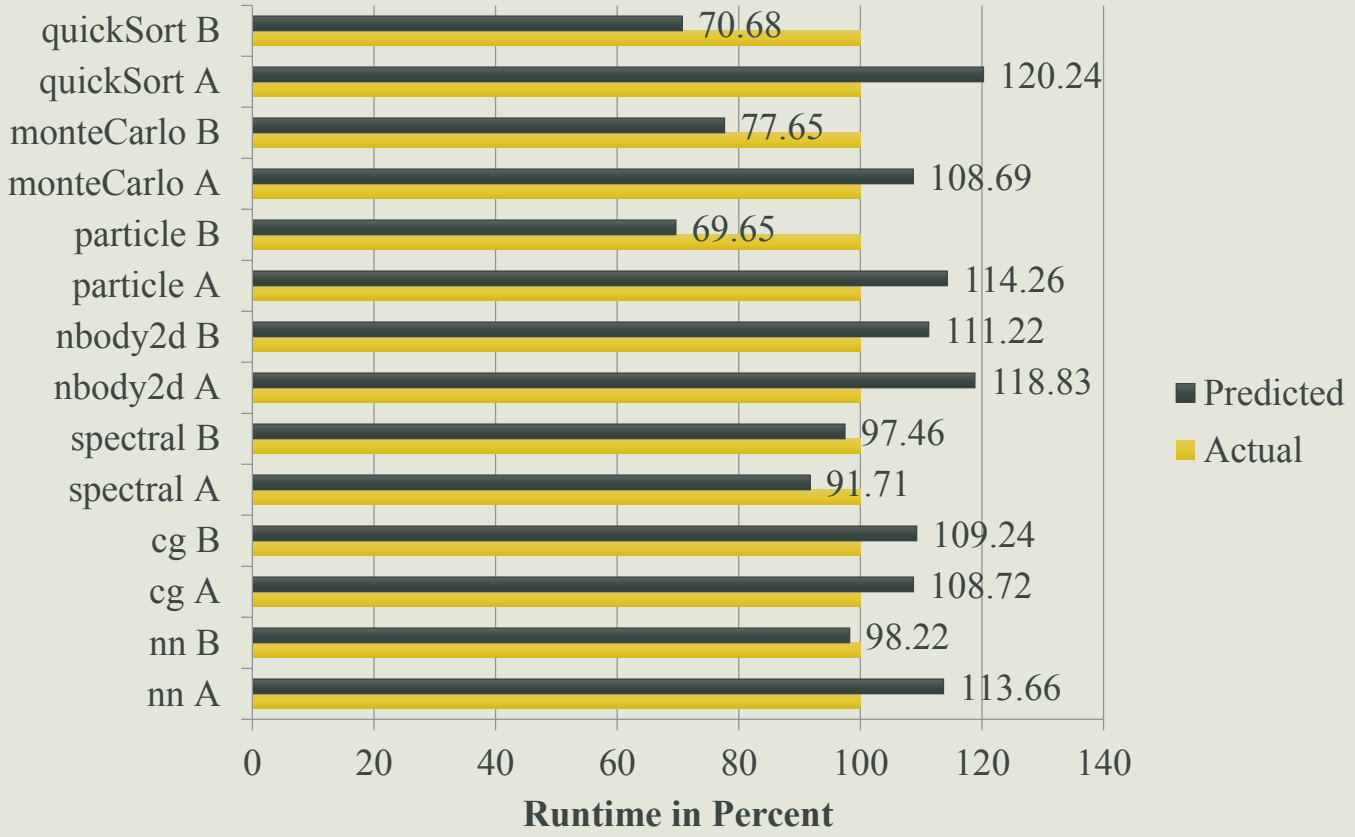
Runtime Estimation Model

- Compute the estimated runtime of the application on a specific machine
- There will be 13 models for each type of machines
- Construct the mathematical models
  - Collect the profiles of each dwarf on machines
  - Use ABC algorithm (meta-heuristic algorithm) to find the appropriate model

Runtime Prediction: Unknown profile

Results on Runtime Prediction

- Sample the profiles of 16 benchmarks
- Compare the predicted runtimes to the actual runtimes
- Error (%)
  - Minimum 8%
  - Maximum 30%



Initial Results

- Testing the model
  - 99% R-squared
  - Less than 10% error (2.28% average)

Input Size (Mb)	Predicted Runtime (s)	Actual Runtime (s)	Error (%)
100	12.06	11.93	1.09
200	19.19	18.79	2.13
300	26.32	25.79	2.06
400	33.45	36.22	7.65
500	40.58	39.95	1.58
600	47.71	46.95	1.62
700	54.84	54.06	1.44
800	61.97	62.23	0.42

Job Scheduling

- Meta-Heuristic Algorithm (Artificial Bee Colony)
- Objective Score

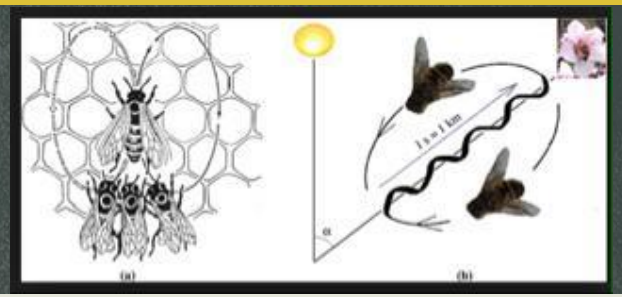
$$Score = (\alpha \times makespan) + (1 - \alpha) \times (Energy)$$

$$makespan = \max(CompletionTime_n + WaitingTime_m)$$

$$Energy = Energy_{idle} + Energy_{switch} + Energy_{Active}$$

- α is the significant level of the objective

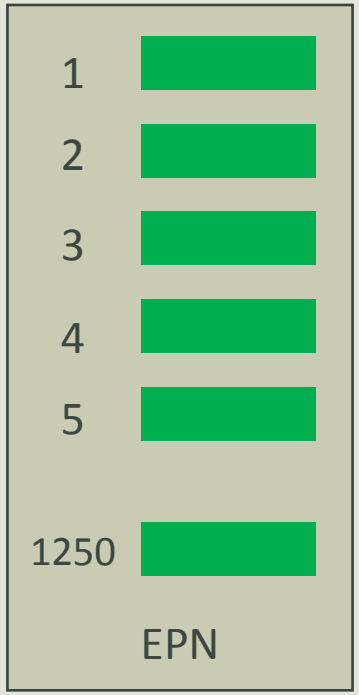
Artificial Bee Colony

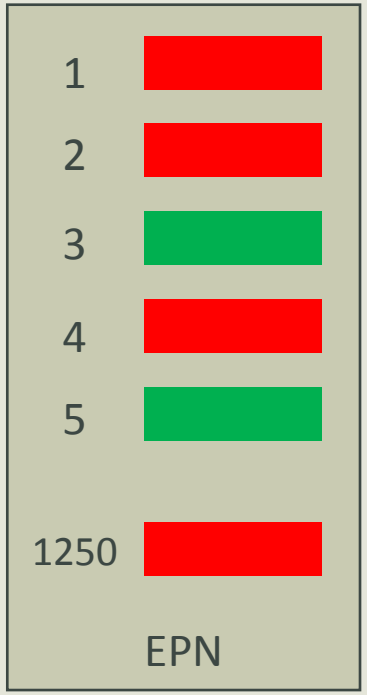


- Mimicking the behaviors of honeybees in finding food sources
- Each feasible solution represents a food source.
  - The quality of a food source is the “fitness value”
  - The process of bee seeking for good food sources is used to fine the optimal solution.
- There are 3 types of computational agents
  - Employed Bee: Investigated the assigned food sources and share information
  - onlooker Bee: make a decision to choose a food source
  - Scout Bee: search randomly for new food sources

Initial Experiments

- 2 situations were simulated
  - Initial Phase: All machines are available
  - Saturated Phase: Only some machine available





Initial Result (1)

- Initial Phase
  - Round Robin has better makespan in initial Phase
  - Use 2 times more energy

Task, Machine	Round-Robin		Artificial Bee Colony	
	Makespan	Energy	Makespan	Energy
2000,1250	417	140622	458	52859
1750,1250	406	123344	426	45727
1500,1250	406	106171	387	38321
1250,1250	210	88138	415	32738
1000,1250	210	70012	387	28074
750,1250	210	52134	300	18184
500,1250	210	34382	253	11539

Initial Result (2)

- Saturated Phase
- ABC show 5-10% improvement in makespan over Round-Robin
- ABC's Energy consumption is 50% lower than Round-Robin

Task, Machine	Round-Robin		Artificial Bee Colony	
	Makespan	Energy	Makespan	Energy
2000,1250	417	140622	458	52859
2000,1000	407	141982	517	54222
2000,750	578	142283	614	55807
2000,500	751	141962	670	57717
2000,250	1422	141101	1350	90865

Work in Progress

- More runtime estimation on several other algorithms
- Fine tuning the scheduling algorithm
  - Increase the speed by simplifying the ABC algorithm
  - Improve the schedule quality
    - Explored other scheduling algorithms used in grid schedulers
  - Prepare for other constraints and policies of the online processing system
    - Data locality
    - hardware requirement
    - Etc.
- Implement the scheduler for ALICE O^2

Initial Outputs

- The work on computing platform assessment has been presented at RT'14 in Nara, Japan.
  - Chapeland S., B. Changaival, and T. Achalakul, "Benchmarks Based on a Pixel Cluster Finder Algorithm for Future ALICE Online Computing Upgrade", The 19th Real-time Conference (RT'14), Nara, Japan, May 2014.

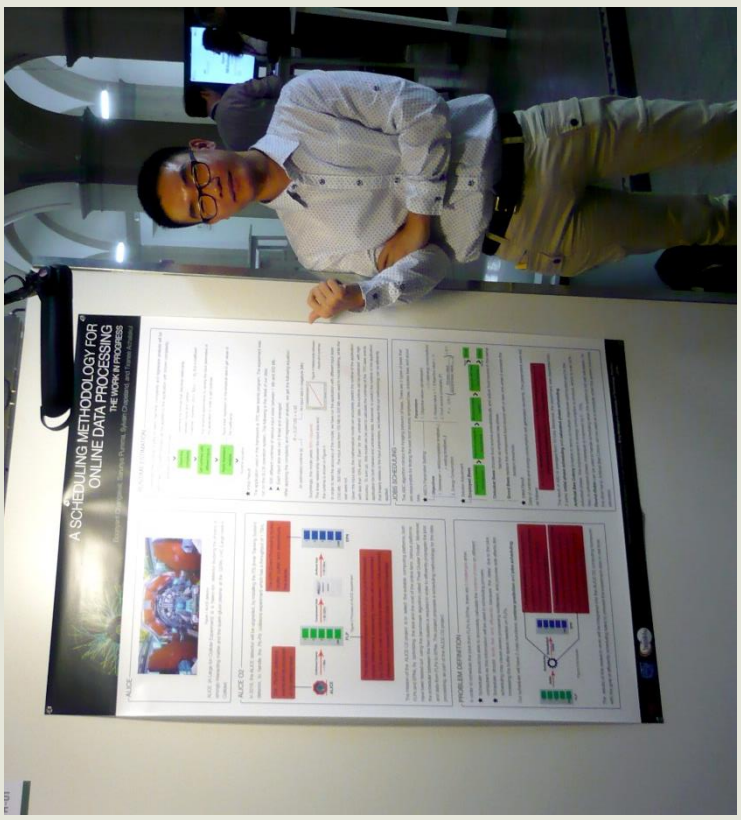
- The work on scheduling framework initial design has been proposed in a poster session of PASC14 in Zurich.
  - Changaival B., S. Pumma, T. Achalakul, and S. Chapeland, "A scheduling Methodology for Online Data Processing" The Platform for Advanced Scientific Computing Conference (PASC14), Zurich, Switzerland, June 2014.

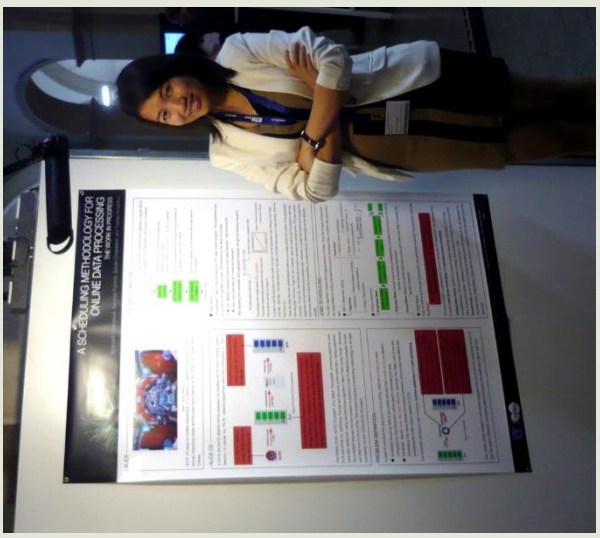
Activities for Summer 2014

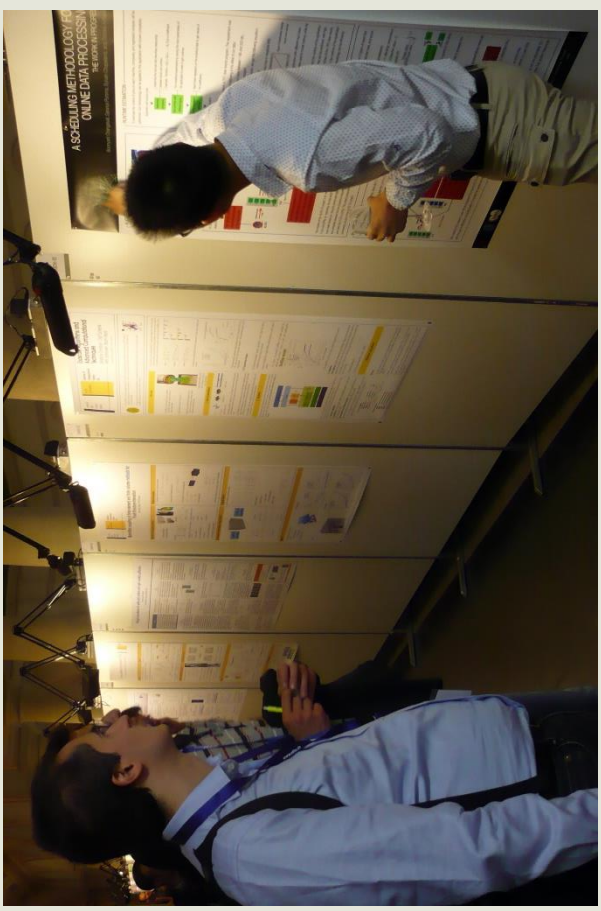
- Mr. Boonyarit Changaival and Ms.Sarunya Pumma delivered the presentation at PASC14 in Zurich.
- Both students is spending 2 months working at CERN, Switzerland.
- Dr.Tiranee Achalakul will visit CERN during July 2014.
- At the end of the internship period, if mature enough, a full paper should be written to describe the first phase of our research works.
- Three more graduate students are being added to the team by the end of the summer.

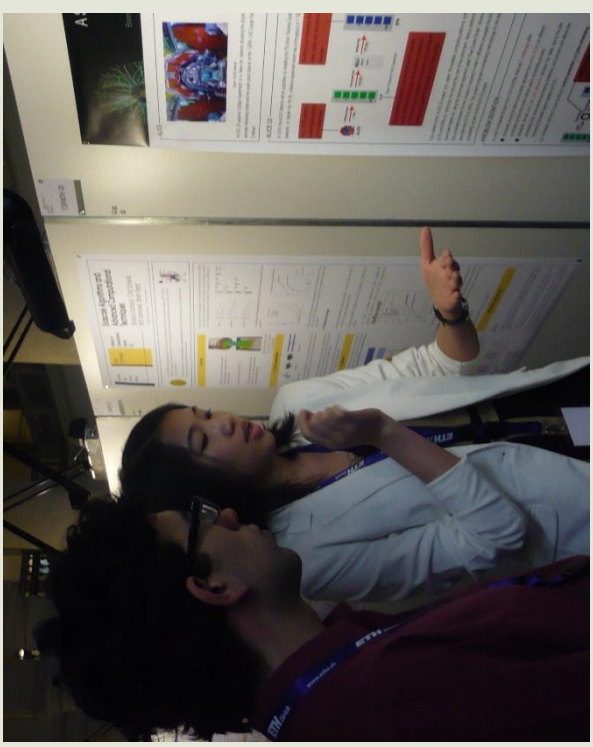
PASC2014 at Zurich, Switzerland

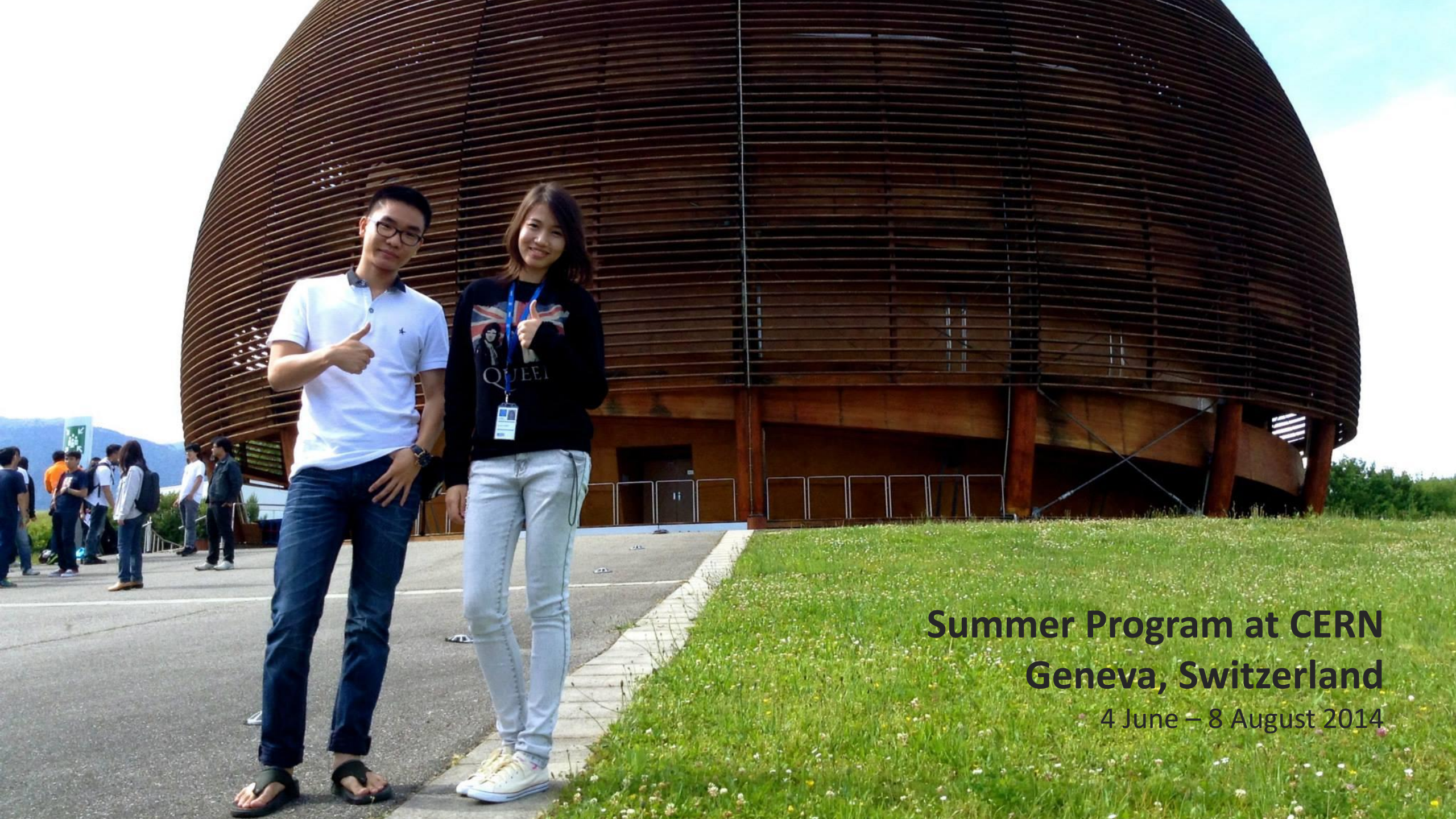
2 – 3 June 2014











Summer Program at CERN
Geneva, Switzerland

4 June – 8 August 2014