Improving System Performance Through Log Analytics, Abnormality Detection, and Task Scheduling (Remotely)

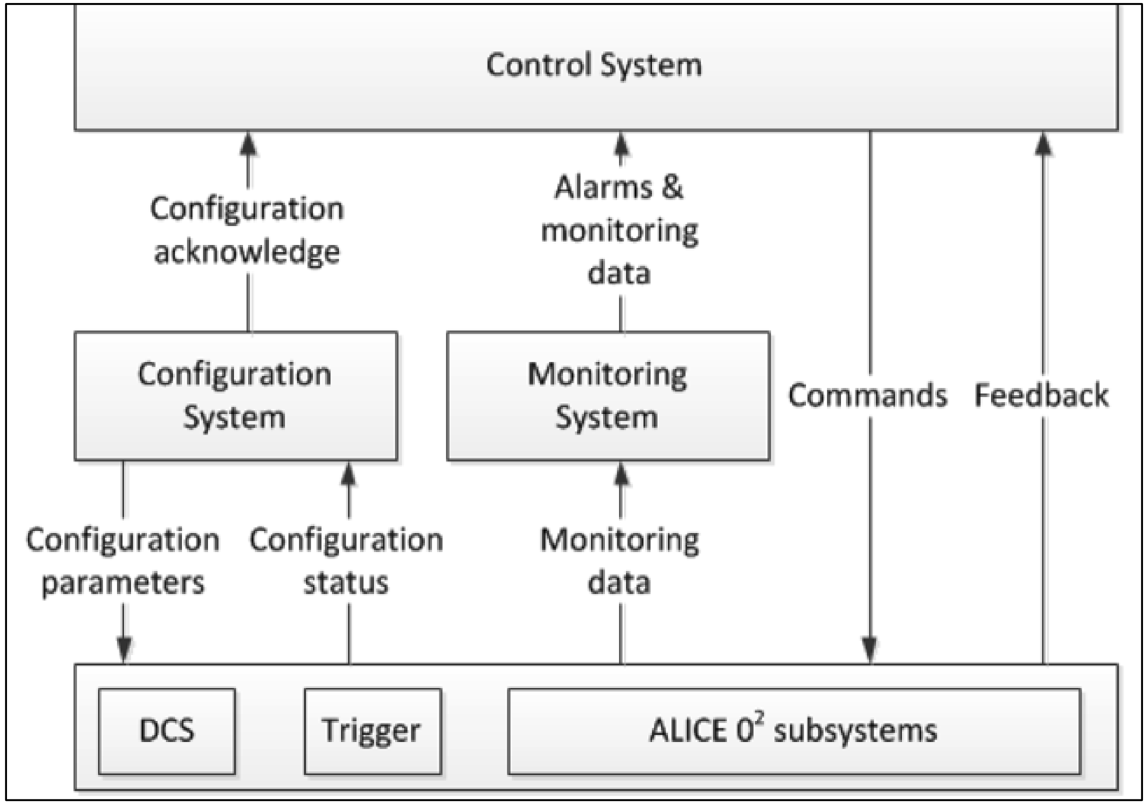
Baramee Sansaengtham

November, 21 2018

Overview Topics

1. Time Series Image Classification for Survival Analysis Using Deep Learning – *Baramee Sansaengtham*
2. Anomaly Detection Using Log Analytics – *Purimpat Cheansunan*
3. Artificial Bee Colony Scheduler - *Ratchapong Krobpan*

Control, Configuration, and Monitoring System (CCM)



Relationship between the CCM components

CCM are components act as a tightly-coupled entity that support users and automate day-to-day operations.

1. **Control system**: responsible for coordinating all the O^2 processes.
2. **Monitoring system**: gathers information from the O^2 system, identifying unusual patterns and raising alarms.

Time Series Image Classification for Survival Analysis Using Deep Learning

Baramee Sansaengtham

November, 21 2018

Outline

- Objective & Scope
- Data Gathering
  1. Monitoring Architecture
  2. Modular Stack
  3. Collected Dataset
  4. Solution: Google Cluster Dataset
- Model Architecture
- Project Schedule
- Future plan

Objective & Scope

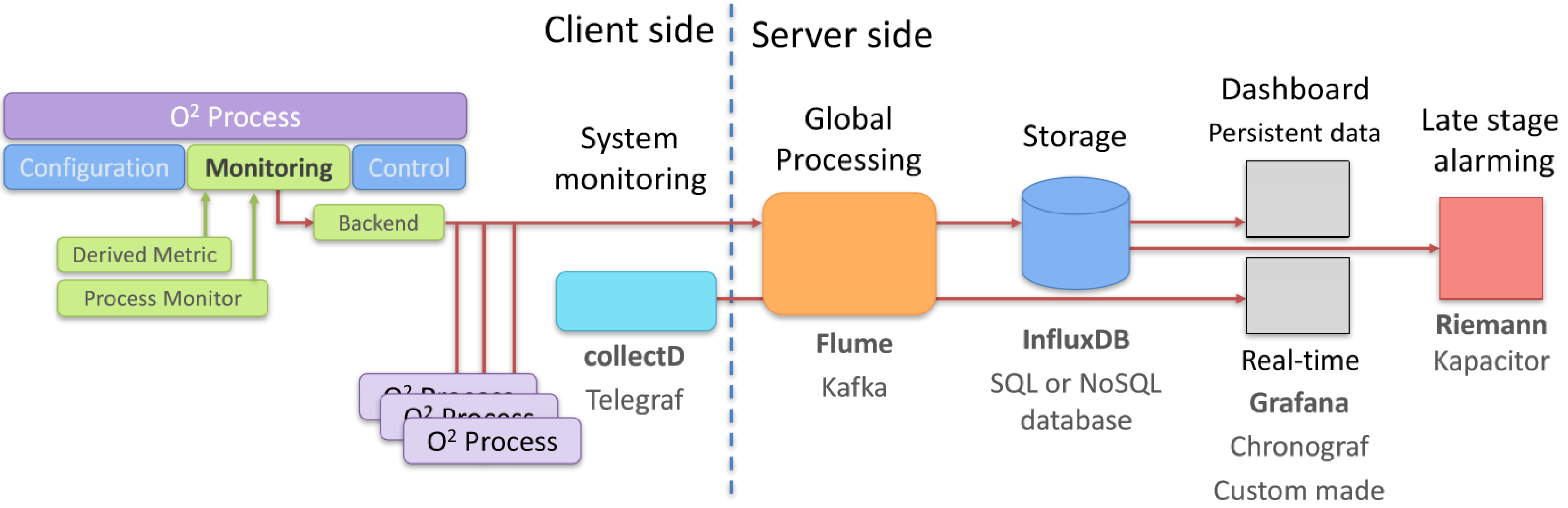
- Proof of Concept:

"To prove the concept of deep learning with predictive maintenance and improve the prediction model with time-series images"

Survival Analysis Scope

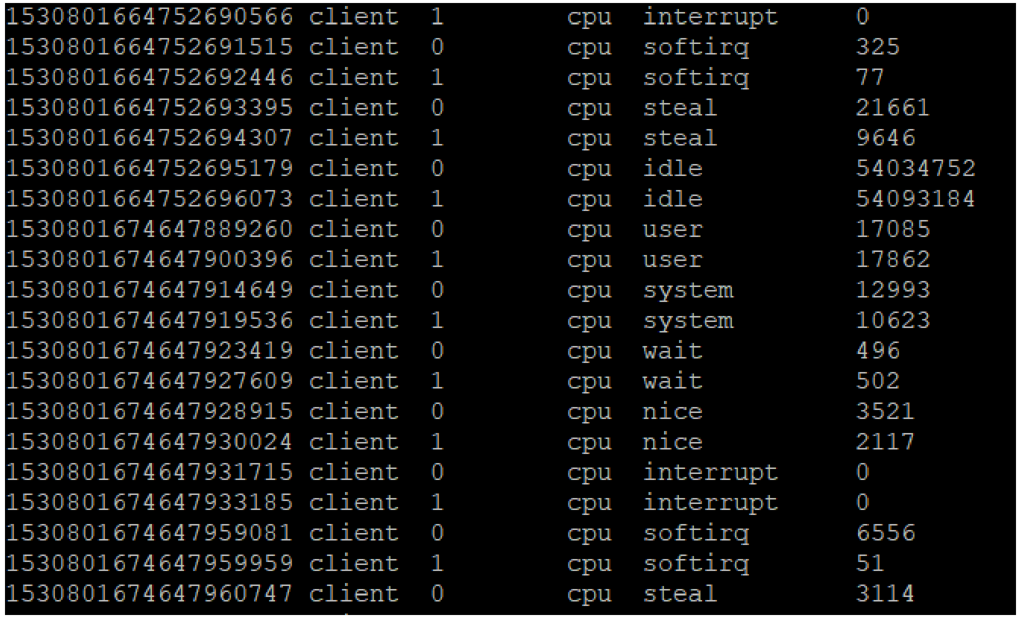
- Predicting the distribution of future time-to-failure (life cycle)
- Transform time-series data into sequential image and feed to the classification model
- The approach can be used to predict failures of any component in many other application domains

Data Gathering - Modularity Architecture



Collected Dataset

- System utilization metrics
- Time-series data
- Possible metrics from Collectd
  1. Disk
  2. Network
  3. CPU
  4. Entropy
  5. Load
  6. Memory
  7. Swap
  8. Uptime
  9. Processes
  10. Sensors



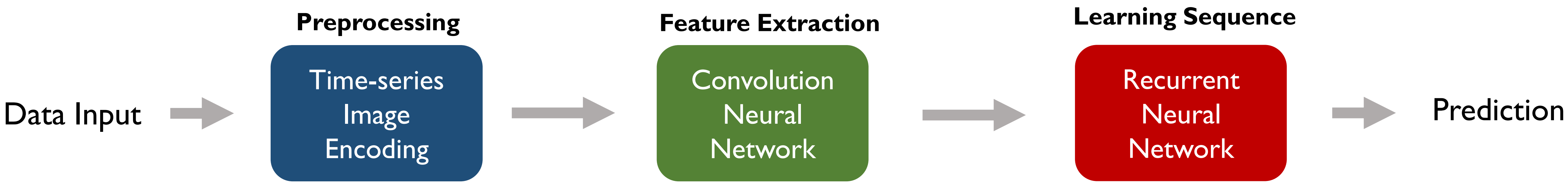
```
1530801664752690566 client  1          cpu  interrupt      0
1530801664752691515 client  0          cpu  softirq        325
1530801664752692446 client  1          cpu  softirq        77
1530801664752693395 client  0          cpu  steal          21661
1530801664752694307 client  1          cpu  steal          9646
1530801664752695179 client  0          cpu  idle           54034752
1530801664752696073 client  1          cpu  idle           54093184
1530801674647889260 client  0          cpu  user           17085
1530801674647900396 client  1          cpu  user           17862
1530801674647914649 client  0          cpu  system         12993
1530801674647919536 client  1          cpu  system         10623
1530801674647923419 client  0          cpu  wait           496
1530801674647927609 client  1          cpu  wait           502
1530801674647928915 client  0          cpu  nice           3521
1530801674647930024 client  1          cpu  nice           2117
1530801674647931715 client  0          cpu  interrupt      0
1530801674647933185 client  1          cpu  interrupt      0
1530801674647959081 client  0          cpu  softirq        6556
1530801674647959959 client  1          cpu  softirq        51
1530801674647960747 client  0          cpu  steal          3114
```

Google Cluster Dataset

- Distributor: Google
- System: 12.5k machines in datacenter
- Data Type:
  1. The semantics, data format, and schema of usage traces of a Google compute cell
  2. Each task is a Linux program that was assigned to the cell by management system
- Duration: Trace represents 29 days from May 2011
- Data size: 41 GB

Advantage	Disadvantage
Plenty of useable metrics	Some missing information
Clear state of the task	
Cleaned data format	
Reliable	

Model Architecture



Preprocessing

Feature Extraction

Learning Sequence

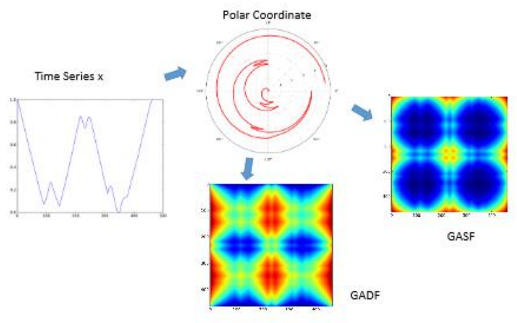
Data Input

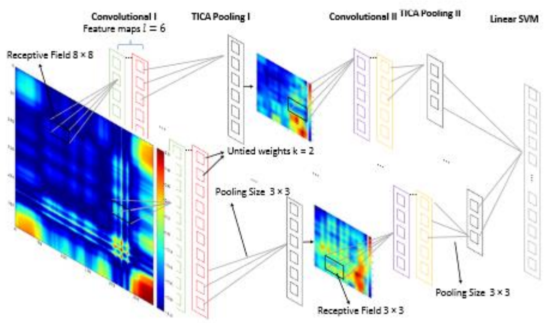
Time-series Image Encoding

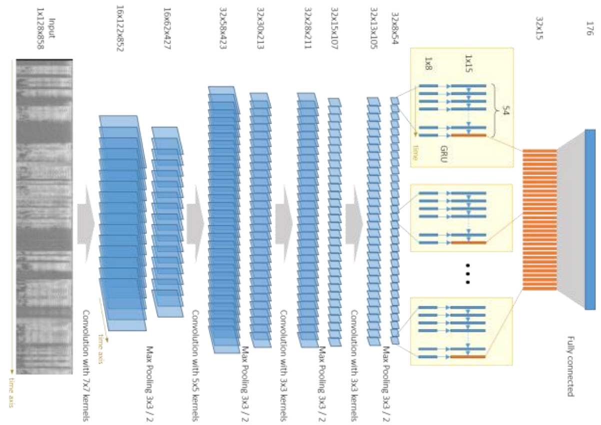
Convolution Neural Network

Recurrent Neural Network

Prediction







Project Schedule

Period	Description
January – May 2018	- Study about tools and monitoring system to gather info data - Research and implement a simple model with example data
June – July 2018	- At CERN, Deploy and test a gathering data tool
August – December 2018	- Review the data to build a predictive models
Until January 2019	- Process the models with the testing data - Write down the result to the paper

Future Plan

- Preprocess and transform data
- Develop the model architecture
- Test and improve the performance of the model
- Apply model with the data from ALICE

Anomaly Detection Using Log Analytics

Purimpat Cheansunan

November, 21 2018

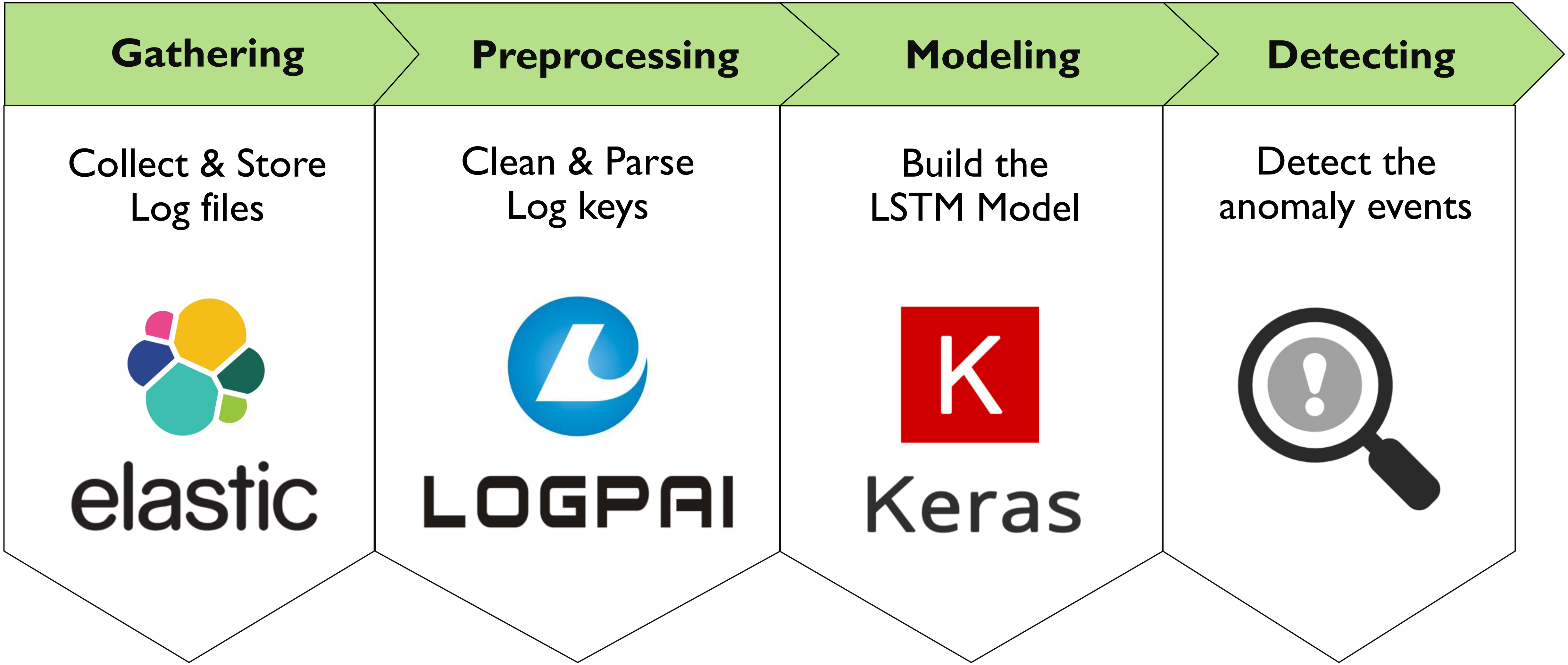
Outline

- Scope & Objective
- Overview architecture
- Project Status
  1. Gathering Phase
  2. Researching Phase
  3. Implementing Phase
- Future plan

Scope & Objective

- To build the AI-based monitoring system for detecting the anomaly event pattern inside the system log message.
- When the system detect the abnormal log pattern, system administrator will be alerted.
- To prevent the occurrence of system failure.

Overview Architecture



Gathering

Collect & Store Log files

Preprocessing

Clean & Parse Log keys

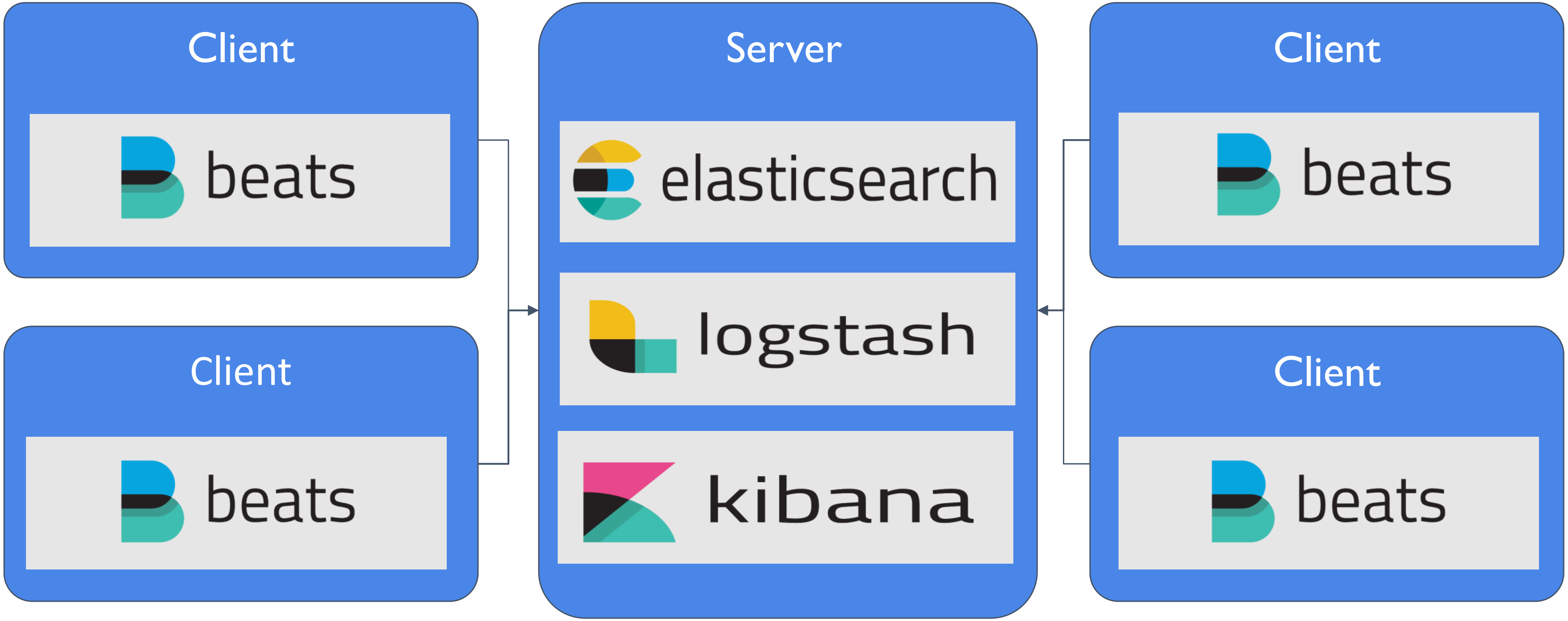
Modeling

Build the LSTM Model

Detecting

Detect the anomaly events

Elastic Stack



Example of Log data from Elastic Stack

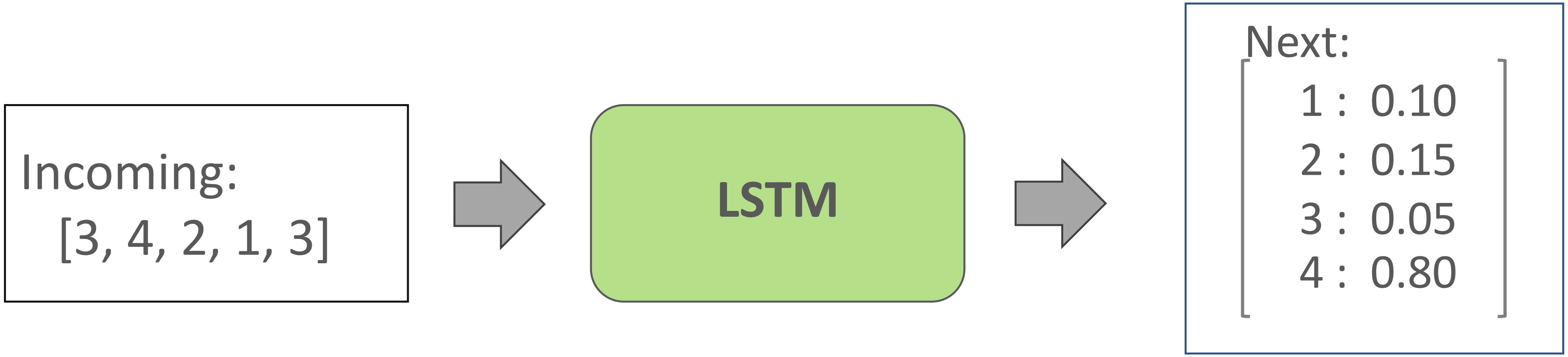
Timestamp	Syslog host	Program	Message
2018-07-05 13:25:22	kmutt-cern-elk-stack-client2	chronyd	Selected source 162.23.41.56
2018-07-05 13:25:23	kmutt-cern-elk-stack-client2	kernel	random: crng init done
2018-07-05 13:25:24	kmutt-cern-elk-stack-client2	systemd	Started Getty on tty1.
2018-07-05 13:25:24	kmutt-cern-elk-stack-client2	systemd	Starting Job spooling tools...

LOGPAI: Log parsing library

Key	Template Message	Example Message
1	* monitoring directory *	32346 monitoring directory `/etc` (2)
2	Starting Session * of user *	Starting Session 14 of user root.
3	* monitoring file *	23497 monitoring file `/etc/services` (4)
4	<info> * Loaded device plugin: * (internal)	<info> [1530788604.4880] Loaded device plugin: NMBridgeDeviceFactory (internal)

Long-Short Term Memory (LSTM)

- Log generation patterns will be learned by LSTM.
- Model will predict what should be the following log from the incoming log sequence.



Incoming:
[3, 4, 2, 1, 3]

LSTM

Next:
1 : 0.10
2 : 0.15
3 : 0.05
4 : 0.80

- Output will be a pairs of log key and their probability to be next log entry.

Future Plan

- Research more methods to compare the results from LSTM.
- Improve the performance of log parser
- Implement Detecting phase

Artificial Bee Colony Scheduler

Ratchapong Krobpan

November, 21 2018

Outline

- Objective & Scope
- Task Summary
- Optimization model
  - Simulation Model
  - Simulation Result
- Future Plan

Objective & Scope

- Build a better scheduler that use the data utilization to analyse task and resources
- Implement the knowledge of ABC analysis to extract some pattern of schedule
- The model will be improved and give a difference scheduler on each time

Task Summary

1. Simulator <done>
   - Simulate VM resource and task status
2. Monitoring <pause>
   - This step will provide total measurement that can use for create Optimization Equation
3. Optimization model
   - Create simulation model
4. Create ABC for this problem
   - Create improve simulation model
5. Performance Proof
   - Compare ABC with normal scheduler

Optimization model

Objective: Minimize run time

Resource	Unit	Task resource usage	Unit
CPU	MHz	CPU	MHz
Memory	Mb	Memory	Mb
Disk	Mb	Disk	Mb
Disk IO	MB/s	Disk IO	MB/s
Network IO	MB/s	Network IO	MB/s
		Run time X core*,**	ms.

* Many process use multiple thread so runtime on single core cpu or multi core cpu may difference.

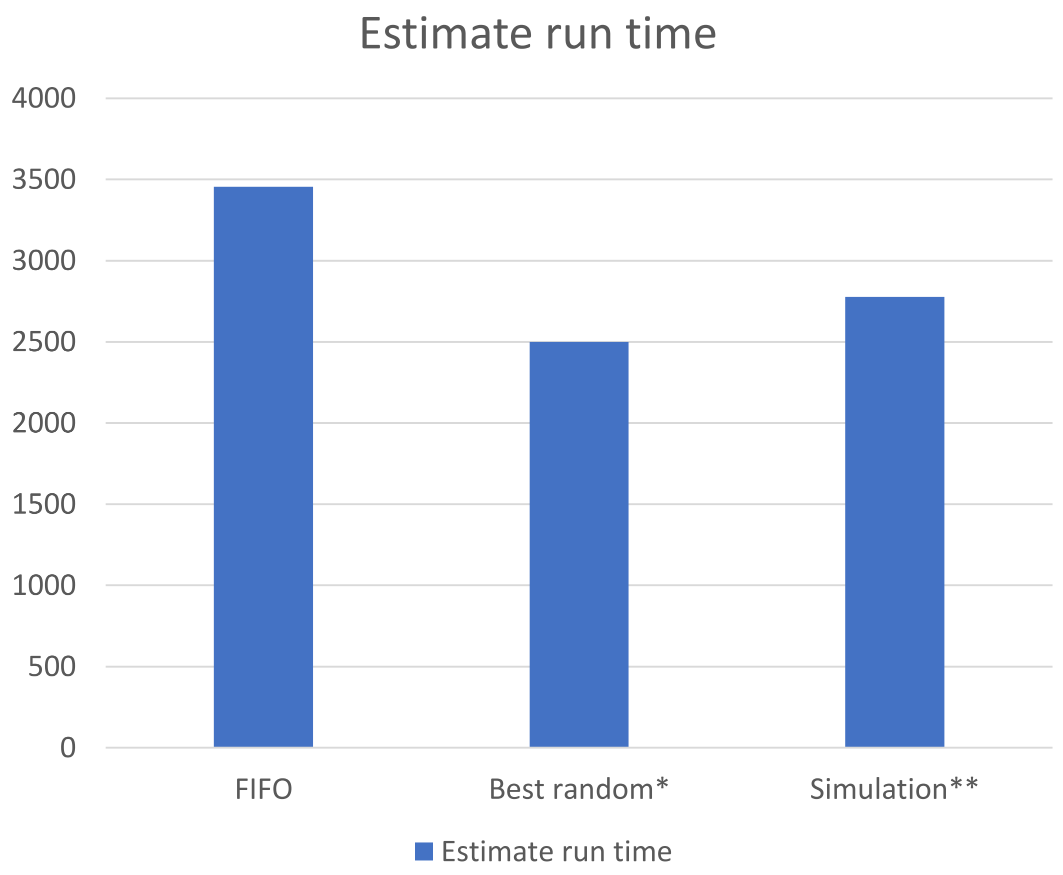
** This average runtime measure by running that process a machine that have more than enough resource

Simulation Model

- List all available resource and incoming task
- For each task
  - Check If this task requirement (pervert task or result from other task)
  - Find and assign this task to the most utilization resource
  - If there is no resource fit for this task just skip this task
- After go through every task if there are still some tasks in the list
  - Shift the time to next (will be) finish tasks
  - Pop finished task and run the loop again

Simulation Result <for now>

Method	Estimate run time	Generate time
FIFO	3456 sec	25 ms
Best random*	2500 sec	1998 ms
Simulation**	2777 sec	22 ms
* Random and test for 2 sec <about 30 samples> ** Resource utilization cap = 1.00		



Future Plan

- Extract knowledge from simulator
  - Make the model can learn and find more effective scheduler
  - Apply ABC to the model
- Performance Proof
  - Simulate and compare result between ABC and other algorithm