Control Configuration Monitoring (CCM)

Khanasin Yamnual
King Mongkut's University of Technology Thonburi
Faculty of Engineering
Department of Computer Engineering
Bangkok, Thailand

Team members

Control

Sirapop Na Ranong

Configuration

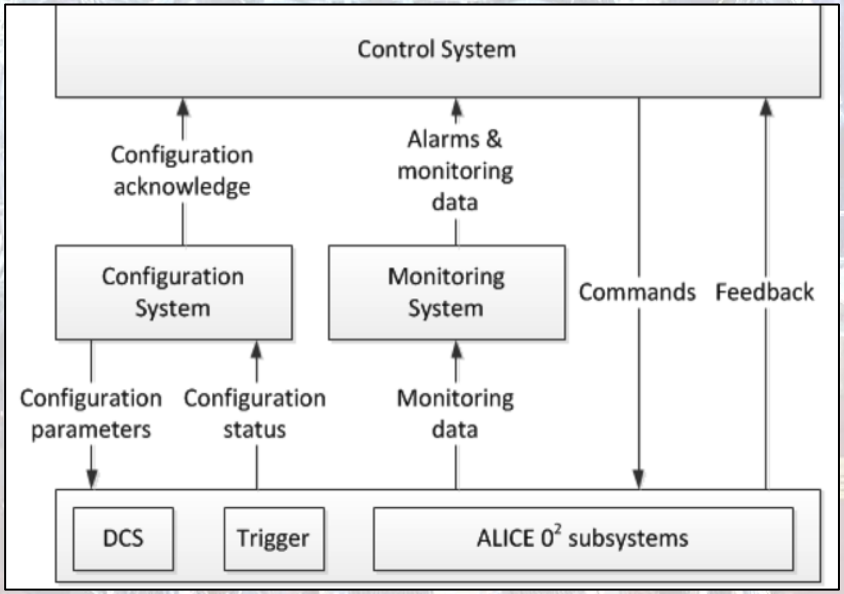
Krittaphat Pugdeethosapol

Monitoring

Khanasin Yamnual

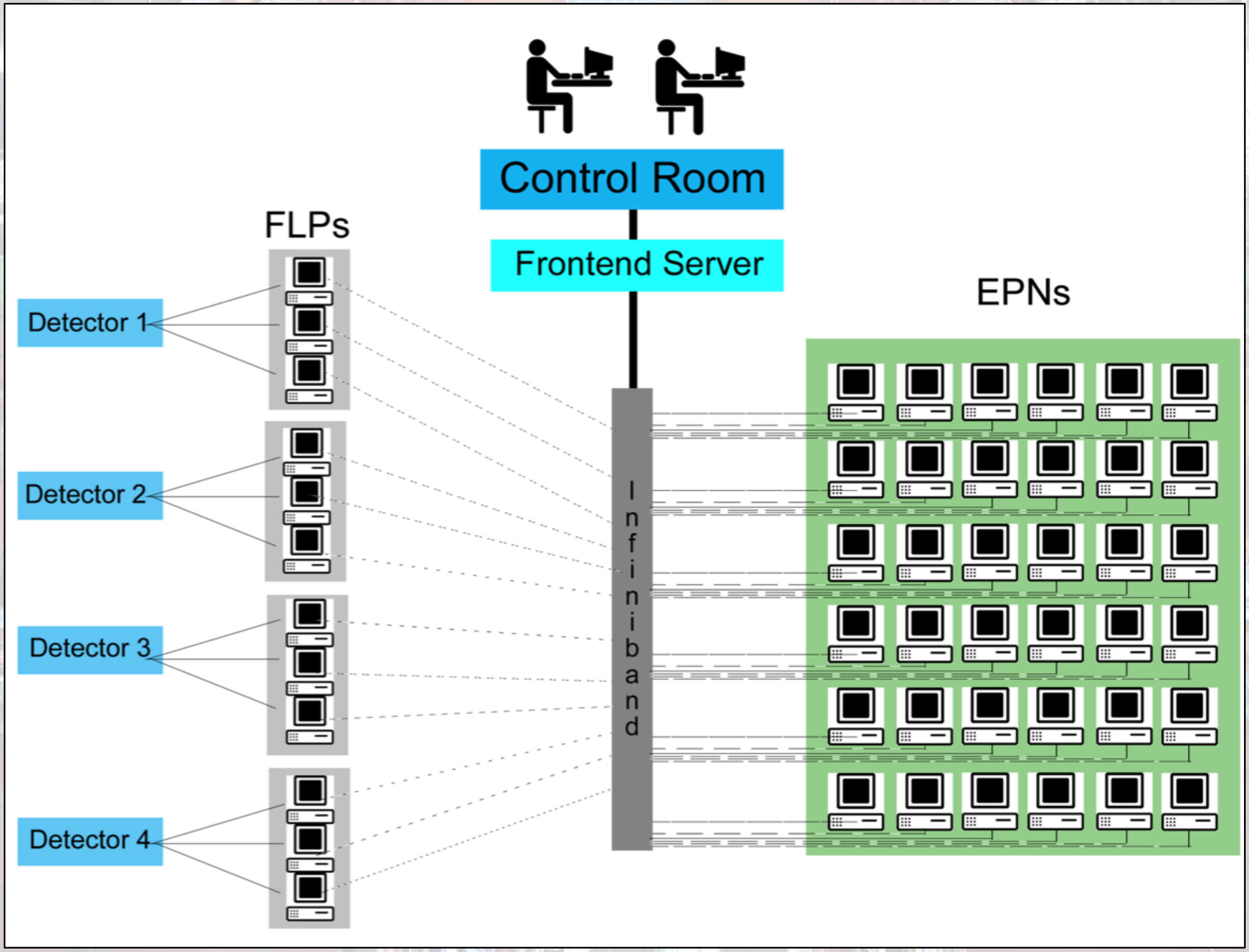
What are CCM?

- support users and automate day-to-day operations.
  - Control system: responsible for coordinating all the O2 processes.
  - Configuration: ensures that both the application and environment parameters are properly set.
  - Monitoring: gathers information from the O2 system, identifying unusual patterns and raising alarms.



Relationship between the CCM components

Architecture



Online environment

Control

- starting and stopping the processes running on the O2.
- includes not only the processes implementing the different functional blocks but also processes providing auxiliary services
- sending commands to running processes
- Typical commands include pausing or resuming the ongoing action
- reacts to internal and external events to achieve a high level of automation
- FSM and Petri-net

FSM

- ***F***inite ***S***tate ***M***achine
- is a tool to model the desired behavior of a sequential system.
- The designer has to develop a finite state model of the system behavior and then designs a circuit that implements this model
- A FSM consists of several ***states***. ***Inputs*** into the machine are combined with the current state of the machine to determine the new state or ***next*** state of the machine.
- Depending on the state of the machine, outputs are generated based on either the state or the state and inputs of the machine.

Petri-net

- Mathematical modeling tools that capture operational dynamics of discrete event systems
- Graphical Representation
- Modeling Language
- State-transition mechanism
- Event Scheduling mechanism

Applications

- Software design
- Workflow management
- Data Analysis
- Reliability Engineering

Software

- STNPlay
- CPNTools (Colored PNs)
- Petri Net Kernel (in Java)
- YASPER (workflow analysis)

Suitable for Modeling

- Concurrency
- Synchronization
- Precedence
- Priority
- Bottom up and top-down modeling

FSM VS. Petri-Net

FSM	Petri-Net
Representation of how one single activity can change its behavior over time, reaction to internally or externally triggered events.	Representation of how multiple activities are coordinated.
Model of discrete behavior, which consists of: a finite number of states, transitions between two of those states, and actions.	Model of showing the interaction between asynchronous processes.

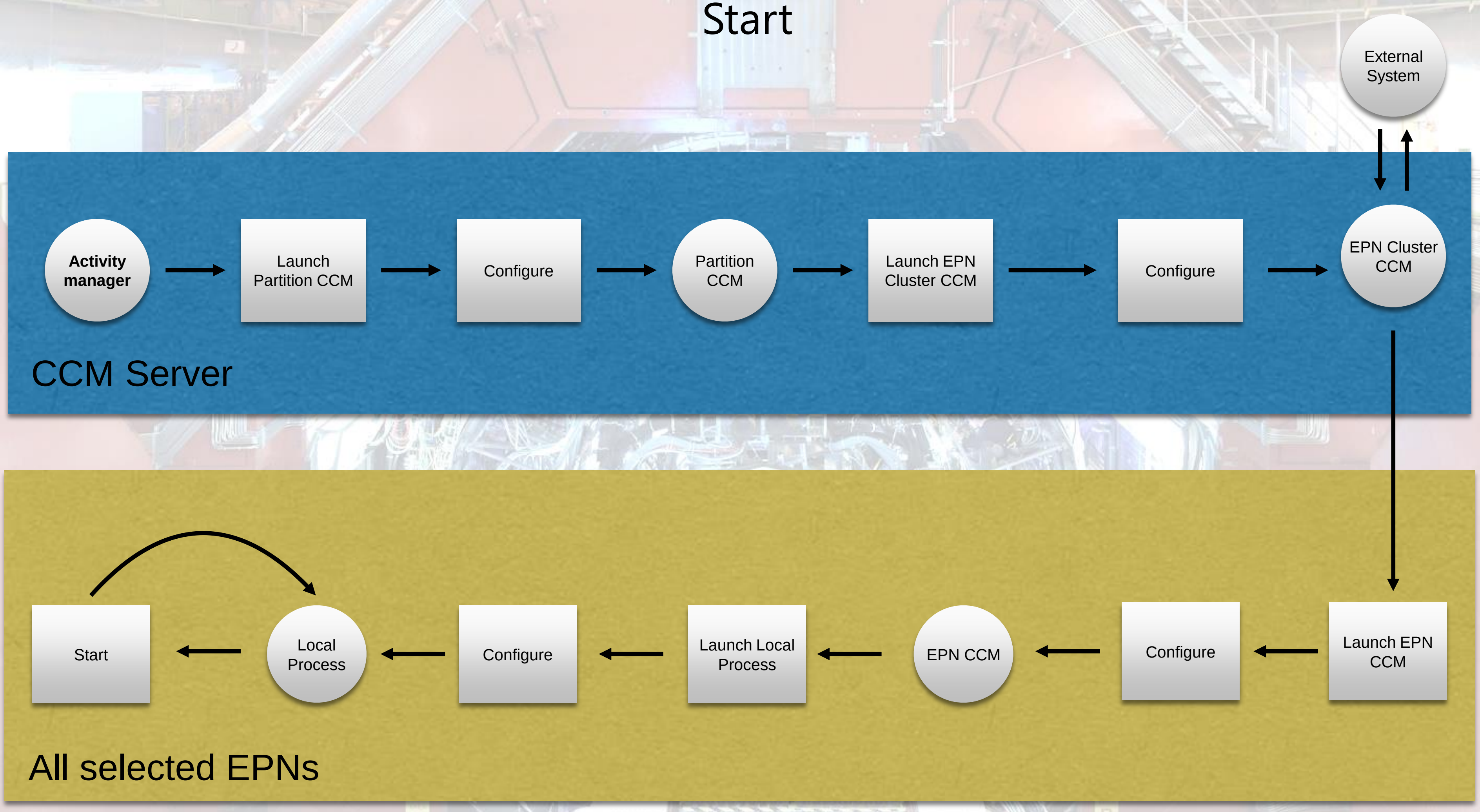
http://en.wikibooks.org/wiki/Embedded_Control_Systems_Design/Finite_State_Machines_and_Petri_Nets

Reason why Petri-Net

- Simplicity of tasks
- Speed gained

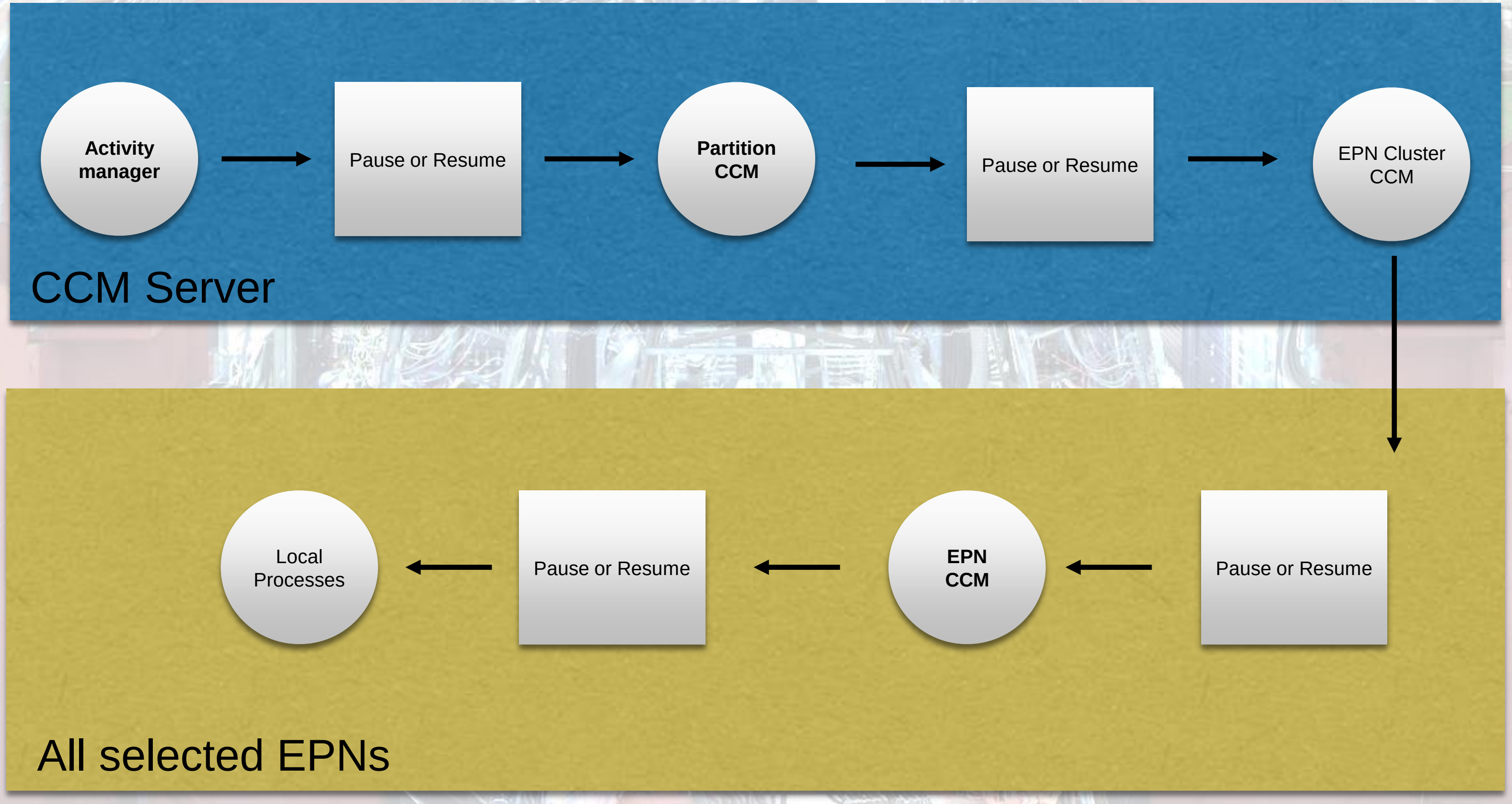
Reconstruction pass

Start



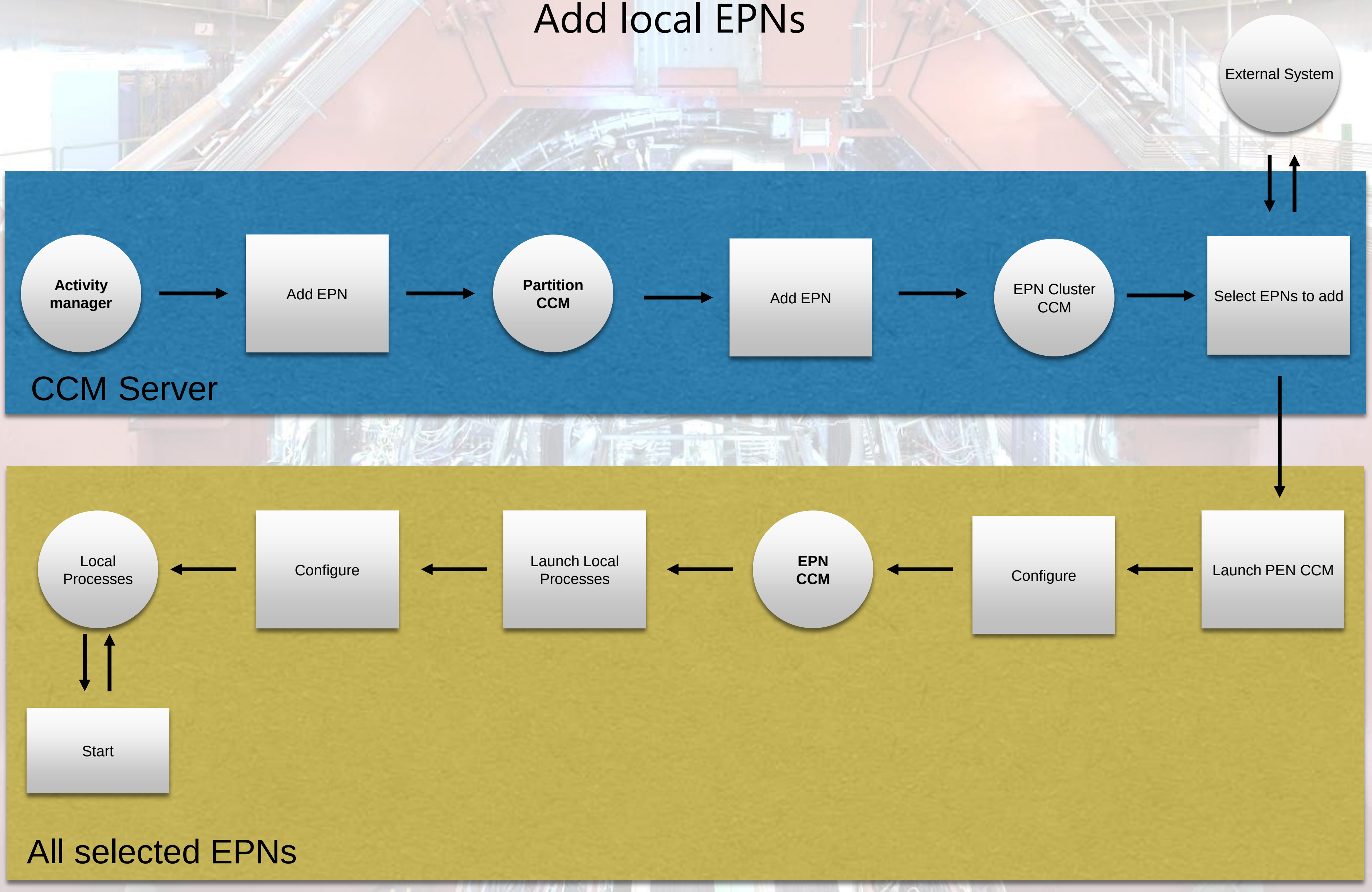
Reconstruction pass

Pause or Resume



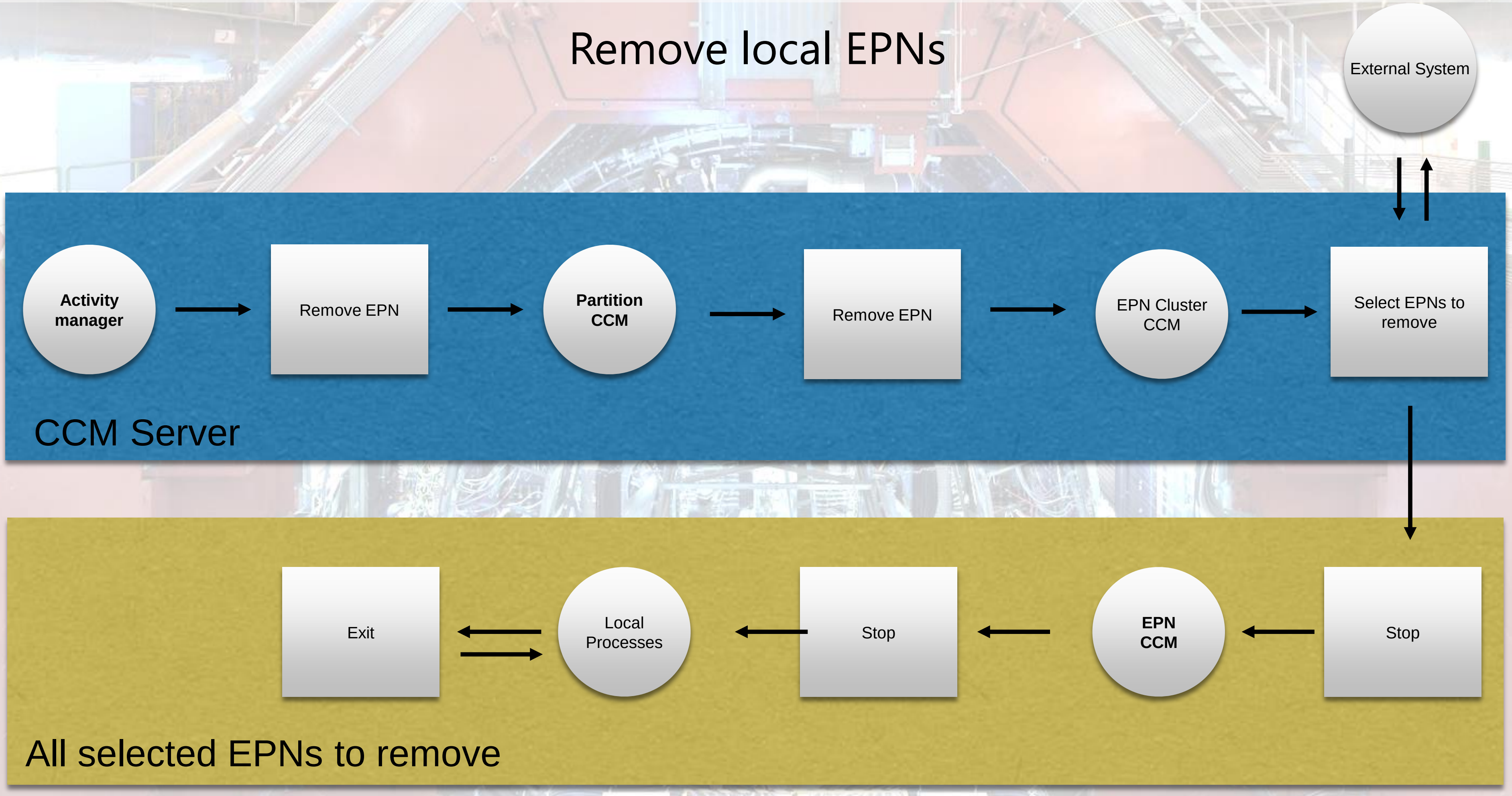
Reconstruction pass

Add local EPNs



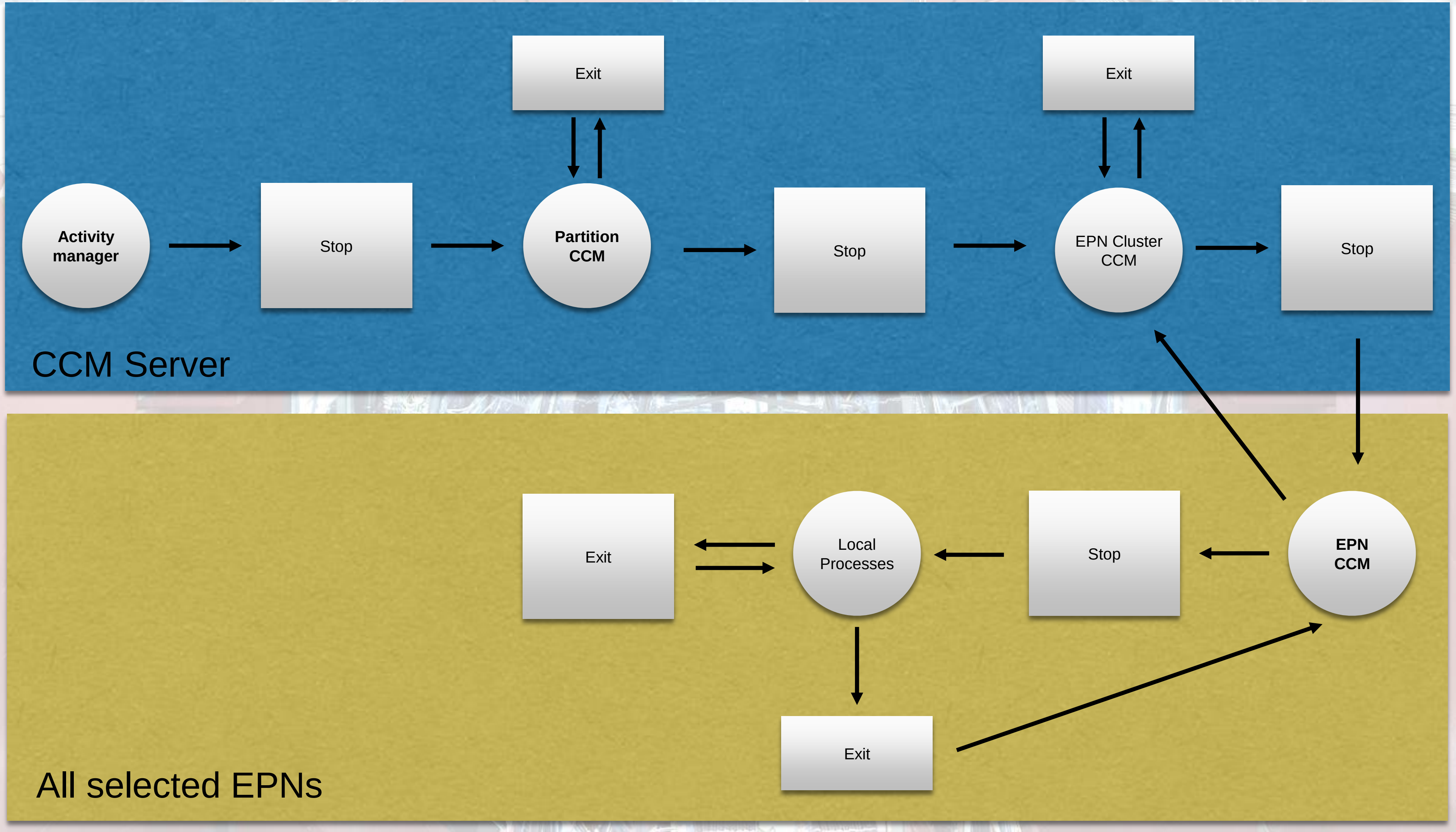
Reconstruction pass

Remove local EPNs



Reconstruction pass

Stop



Control Test

- Objectives
  - to find the method to pause/resume and terminate within a small window of time as well as the overhead associated with it
- Equipment

  4 Laptops (CPU: Intel Centrino, Ram: 1 GB, OS: Cern Centos 7)

  1 switch

- Experiment: start/pause/resume/terminate a process on an individual node.
  - Send a command to start that particular process.
  - Send a command to pause that particular process.
  - Send a command to resume that particular process.
  - Send a command to terminate that particular process.

Test result (Start process)

Start process

Number of node(s)	Observation1 Time used	Observation2 Time used	Observation3 Time used	Observation4 Time used	Observation5 Time used	Average Time used
1 node	0.435s	0.453s	0.428s	0.434s	0.48s	0.446s
2 nodes	0.988s	0.99s	0.949s	0.977s	0.969s	0.975s
3 nodes	1.458s	1.466s	1.438s	1.46s	1.389s	1.442s

Pause process

Number of node(s)	Observation1 Time used	Observation2 Time used	Observation3 Time used	Observation4 Time used	Observation5 Time used	Average Time used
1 node	0.439s	0.473s	0.42s	0.475s	0.491s	0.46s

Resume process

Number of node(s)	Observation1 Time used	Observation2 Time used	Observation3 Time used	Observation4 Time used	Observation5 Time used	Average Time used
1 node	0.448s	0.385s	0.515s	0.432s	0.445s	0.445s

Stop process

Number of node(s)	Observation1 Time used	Observation2 Time used	Observation3 Time used	Observation4 Time used	Observation5 Time used	Average Time used
1 node	0.451s	0.451s	0.451s	0.513s	0.426s	0.458s

Plan to do

- Implement the state machine of cases
- Test the state machine in simulated environment
- Do mathematical proof as well as optimize the state machine

Configuration

- distributing the configuration.
- 2 Types of configuration;
  - Static configuration
  - Dynamic configuration
- Also for software installation and configuration.

Dynamic Configuration

- Parameters Configuration

name	value
checkCompletionSleepTime	1
edmEnabled	1
endOfDataTimeout	10
hltEnabled	1
ldcSocketSize	0
localRecordingDevice	/dev/null
logLevel	10
maxBursts	0
maxBytes	0
maxErrors	10
maxEvents	0
maxEventSize	2000000
maxFileSize	0
monitorEnableFlag	1
pagedDataFlag	1
phaseTimeoutLimit	30
recMaskSleepCntLimit	500
recMaskSleepTime	0
recorderSleepTime	0
sodEodEnabled	1
sorSeparateFile	0
startOfDataTimeout	10

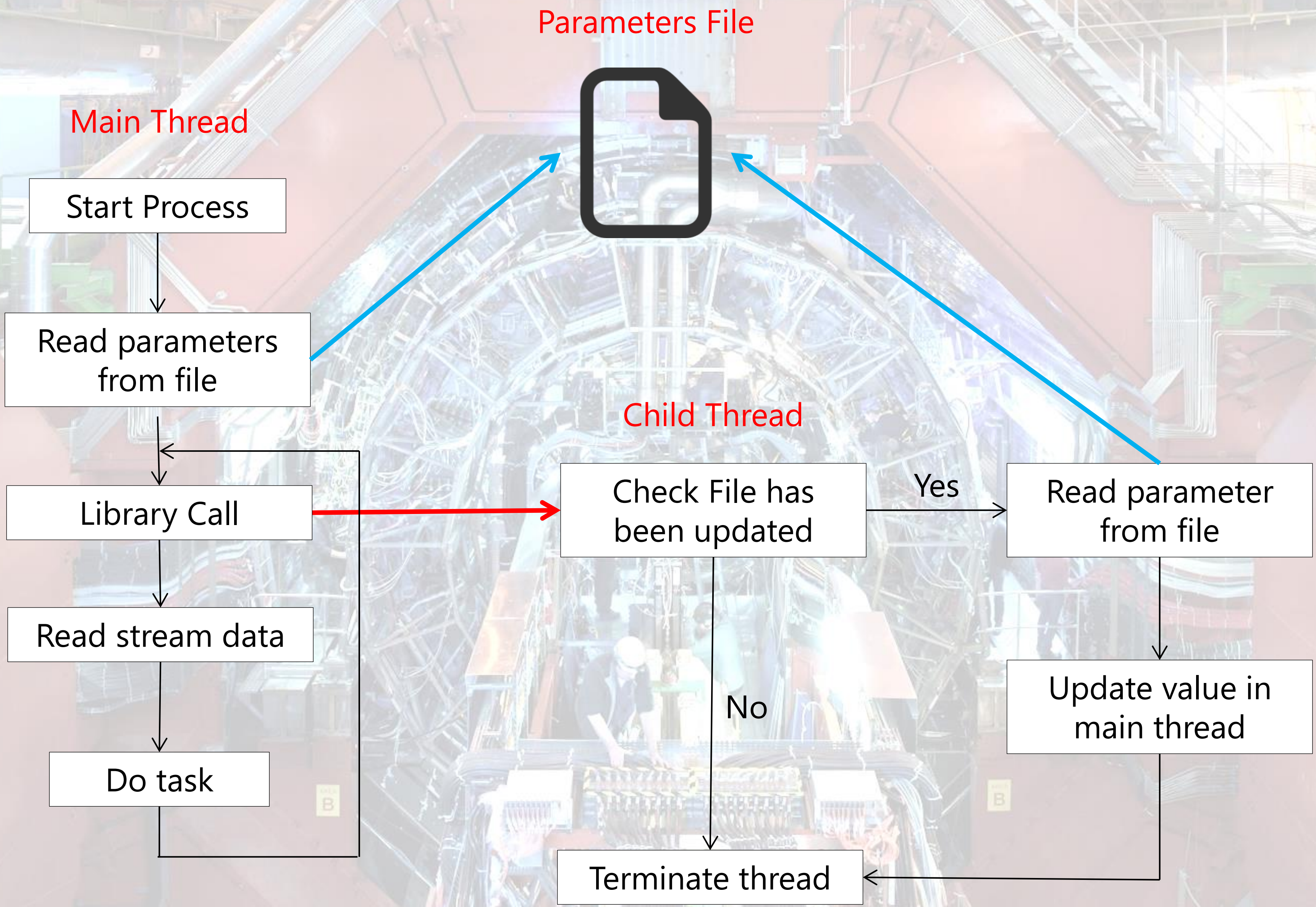
Process

```
Main()
{
    Open file parameter for specific process;
    Read parameter from file;
    While (...............)
    {
        Library Call (...............);
        Read stream data;
        Do task;
    }
}
```

Library Call (parameters that want to change)

- Separate the thread to check if the file has been updated.
- If the file has been updated, it will read the new parameter and update the value then kill thread.
- If the file has not been update, it will do nothing and kill thread.

Flow Chart


Parameters File
Main Thread
Start Process
Read parameters from file
Library Call
Read stream data
Do task
Child Thread
Check File has been updated
Yes
Read parameter from file
Update value in main thread
No
Terminate thread

Parameter File

- Will be unique for each process (can use process ID as file name)
  - Automatically generate the parameter file when start the process
  - File will be deleted when process has been terminated

Plan to do

- Start with one process in one computer (To test that library is worked)
  - Many process in one computer
  - One process in many computer
  - Many process in many computer
- Experiment

  Collect the time that use to re-configure the parameter for each Test

Monitoring

- responsible for processing heartbeat data in quasi real time in order to trigger alerts or take automatic corrective actions.
- responsible for aggregating monitoring data streams and persistently storing the relevant metrics.
- MonALISA and Zabbix

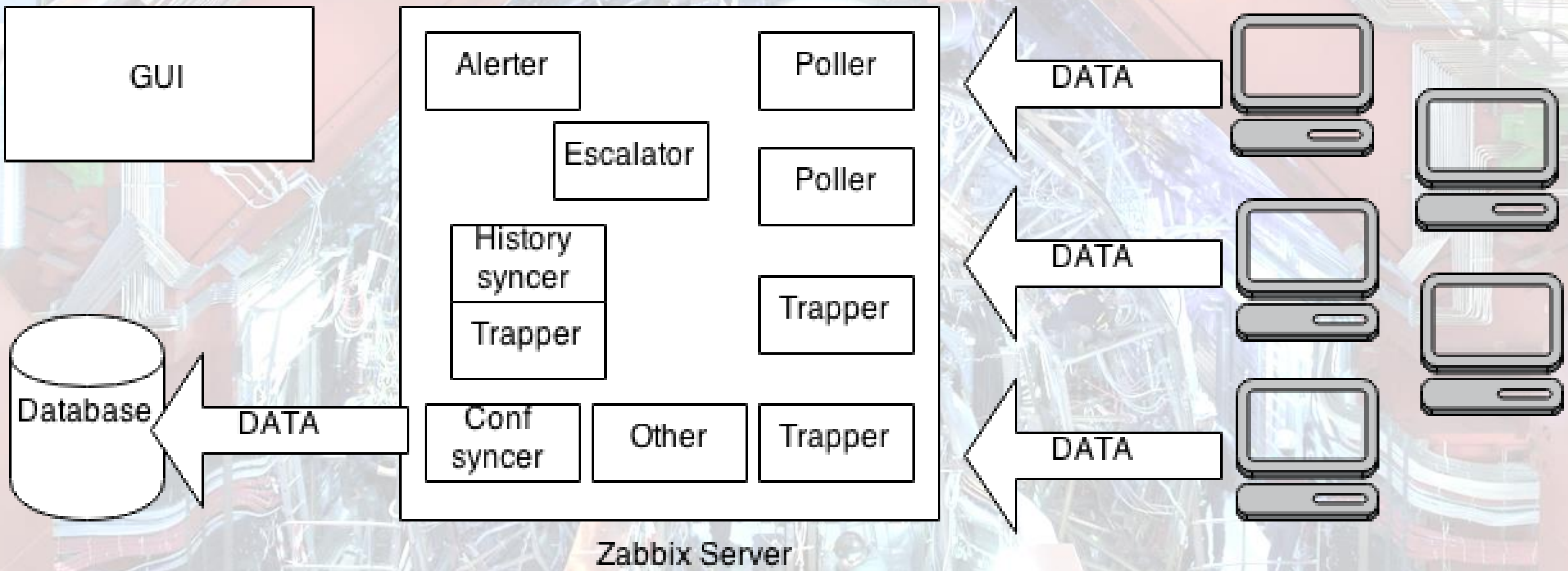
MonALISA

- is Java-based set of distributed, self-describing services.
- Offers the infrastructure to collect any type of information
- Can process data in near real time.
- Take automated decisions and perform actions based on it.
- ALICE uses this tool for monitoring online reconstruction.
- Is simple to install, configure and use.

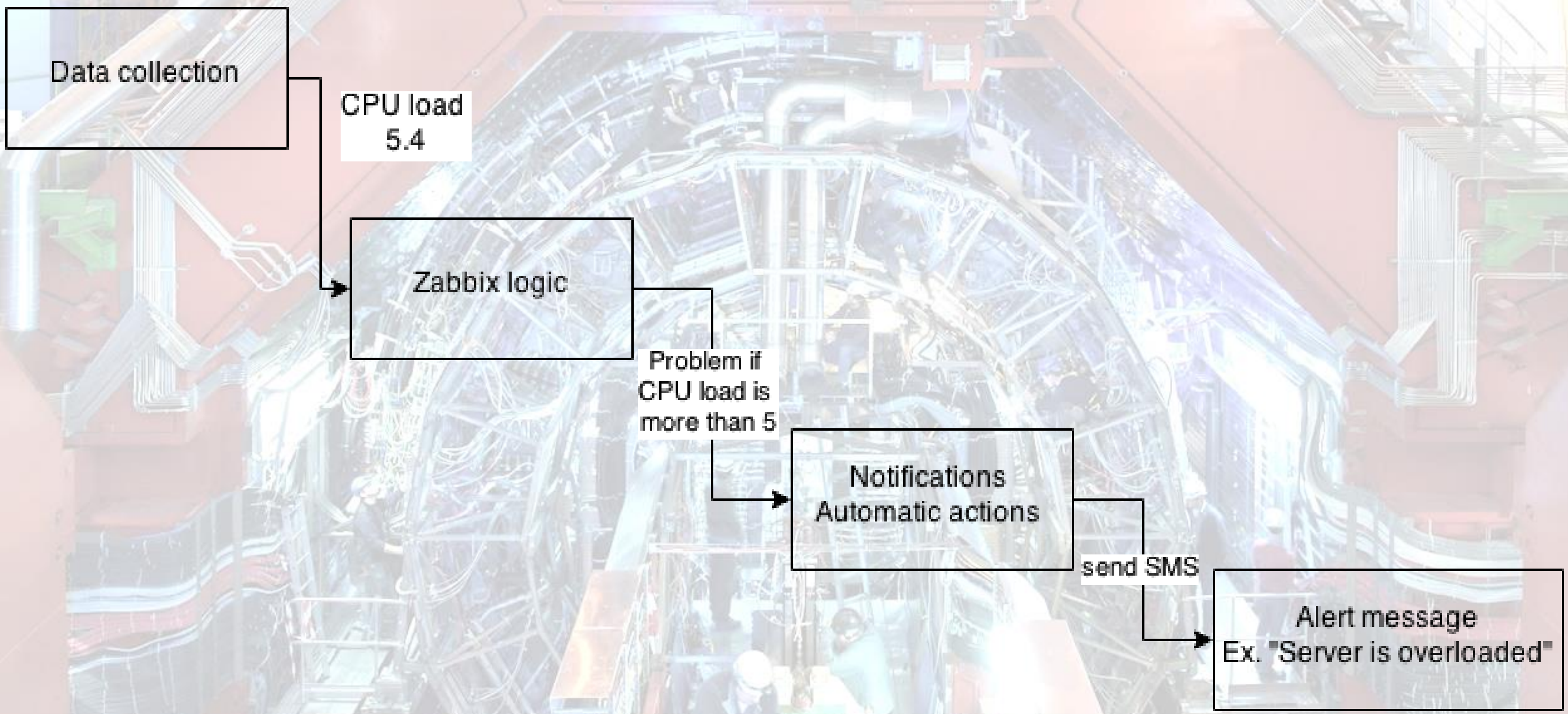
Zabbix

- is an open-source software for monitoring of IT infrastructure.
- is used to monitor numerous parameters of a network and the health and integrity of servers.
- Provide flexible notification mechanism that allows user to configure e-mail based alerts for virtually any event.
- Goals are to identify and fix problems early and to measure and analyze availability and performance

Basic data flow



How Zabbix works?



Conclusion

- Control
  - Petri-Net
- Configuration
  - Dynamic configuration
- Monitoring
  - Find best monitoring tool

References

- Petri Nets: An Overview, Renata Kopach- Konrad
  link: http://ww2.it.nuigalway.ie/staff/pbigioi/ct101/CT101_FiniteStateMachines.ppt
- Finite State Machines in Games, Jarret Raim
  link: http://www.cse.lehigh.edu/~munoz/CSE497/classes/FSM_In_Games.ppt
- Finite State Machines, Mike Chen
  link: http://www.cs.sjsu.edu/faculty/lee/cs147/Finite%20State%20Machines.ppt
- The Petri Net Method, Dr Chris Ling
  link: http://www.utdallas.edu/~gupta/courses/semath/petri.ppt
- Petri Net (lecture10), CES808
  link: http://www.cse.msu.edu/~cse808/note/lecture10.ppt
- An Introduction to Petri Nets, Marjan Sirjani
  link: http://ece.ut.ac.ir/Classpages/S86/ECE658/slides/Petri.ppt
- Finite State Machines, Gaetano Borriello and Randy H. Katzl
  link: http://vada.skku.ac.kr/ClassInfo/digital-logic/zhou/07-FSM.ppt
- Zabbix
  link: http://www.slideshare.net/psihius/zabbix-5713234
- Zabbix, Alexei Vladishev
  link: http://www.slideshare.net/xsbr/alexei-vladishev-zabbixperformancetuning