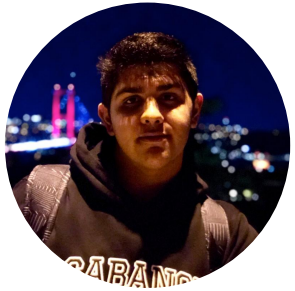
Real-Time Supercomputer Monitoring

ENS491/492 - Presentation

Ali Eren Ak
Computer Science and Engineering



Muhammed Orhun Gale
Computer Science and Engineering

Deniz Batu
Computer Science and Engineering

Problem Definition

- Monitoring High Performance Computers(HPC) is an important research topic
- Bottleneck detection is necessary to improve such systems
- Detection is not a straight-forward task unlike commercial computers
- Current monitoring solutions are either create heavy overheads that are not acceptable for most HPCs in operation or are not real-time

Motivation and Goal

- This project aims to create a light-weight, real-time monitoring tool that can be used to detect bottlenecks and monitor various HPC systems
- A digital twin called SuperTwin was aimed to be created for this goal
- SuperTwin is a reconstruction of an HPC system with the necessary tools to monitor and communicate with the system
- SuperTwin was aimed to have the facilities to let users choose which metrics to monitor and visually inspect statistics and configure to system for performance optimizations in real-time

SuperTwin

- Digital Twin Description Language for Supercomputers
- Data Collection Tools
  - Performance Co-Pilot (PCP)
  - HPCToolkit
    - ReuseTracker
    - ComDetective
- Supercomputer Monitoring Tool (**SuperTwin**)
- Integration, Deployment and Web Application

SuperTwin Description Language

- Digital twin of a physical device allows continuous, structured data reception from the actual device using Digital Twin Description (DTD)
- **SuperTwin Description Language**
  - Based on JSON-LD and Azure Digital Twin Description Language (DTDL) ontology
  - Captures hierarchical structure of the Supercomputers → CPUs, GPUs, Caches etc.
  - Represents all components recursively → each component also a Digital Twin

Property	*Description*
@type	**Interface**
@id	Unique identifier within digital twin for interface
contents	Data type, a set of Telemetry, Properties, Commands, Relationships, Components
displayName	Name to be displayed when instantiated
@type	**Telemetry**
@id	Unique identifier within digital twin for this telemetry instance
name	Programming name, to be referred in queries
schema	Data type
@type	**Property**
@id	Unique identifier within digital twin for this property instance
name	Programming name, to be referred in queries
schema	Data type
@type	**Relationship**
@id	Unique identifier within digital twin for this relationship
name	Programming name, to be referred in queries
properties	Data type, a set of properties

Table 1: Property types that are defined in the Supercomputer DTDL.

Data Collection Tools - PCP

- **Performance Co-Pilot (PCP)**
  - A system performance analysis toolkit
  - Enables real-time system-level data gathering
- Two main units:
  - Performance Metrics Domain Agents (PMDA)
    - Data collection agent → listens specific system-level data sources to gather data
  - Performance Metrics Collection Daemons (PMCD)
    - Collects data from PMDAs → Sends to a time-series database

Data Collection Tools - PCP

- Other important units:
  - Performance Metric Name Space (PMNS) → Structured Data Collection
  - Performance Metrics Inference Engine (PMIE) → Automated Reasoning
- SuperTwin configures following PMDAs:
  - perfevent PMDA
  - proc PMDA
  - linux PMDA
  - lmsensors PMDA

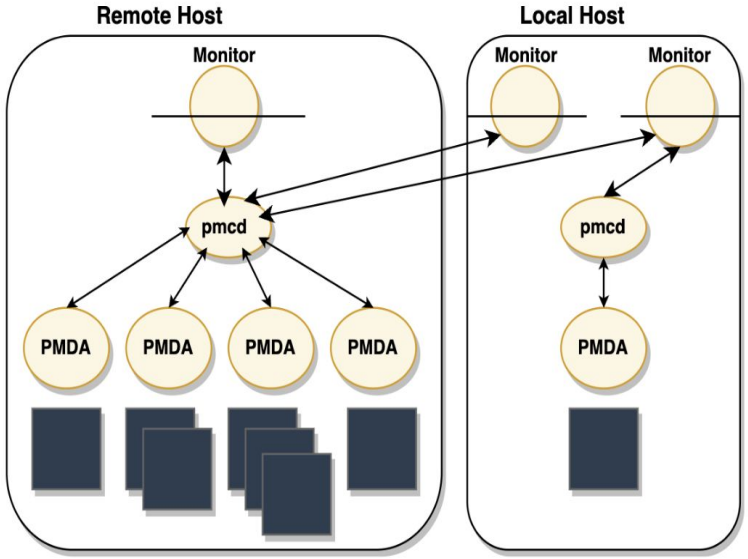


Figure 1: PCP's remote data collection structure

Data Collection Tools - HPCToolkit

- **Inter-thread Communication**
- Communication among threads as the transfer of cache lines across different CPU cores due to cache coherence protocol in a shared-memory system is called inter-thread communication.
- These communications slow a the execution of a program down
- Although it is inevitable in multi-threaded applications, minimizing the frequency of it increases performance

Data Collection Tools - HPCToolkit

- **Performance Monitoring Units (PMU)**
- PMUs are hardware memory-access event counters that can sample and communicate the effective address involved in an event with relevant information
- Intel's Precise Event Based Sampling(PEBS) technology offers a way of configuring and using the PMUs
- They can be used to extract a variable's memory address when a threshold like 10000 loads exceed

Data Collection Tools - HPCToolkit

- **Debug Registers**
- Debug registers are hardware units that are part of processors that can be armed with a memory address to activate a trap in case that memory is accessed by the processor
- When a trap is activated, the address used in arming the trap is being used by the processor/application
- A debug register can hold 8 bytes, processors have 4 debug registers, a cache line is usually 64 bytes so, half of a cache line can be monitored if all debug registers are used (4 x 8 / 64)

Data Collection Tools - ReuseTracker

- Based on “ReuseTracker: Fast Yet Accurate Multicore Reuse Distance Analyzer” paper.
- Aim is to detect reuse distance of memory access of running job in the cluster.
- PMU sampling to sample memory accesses and debug register to trap reuses is used.
- For memory access: a, b, c and d

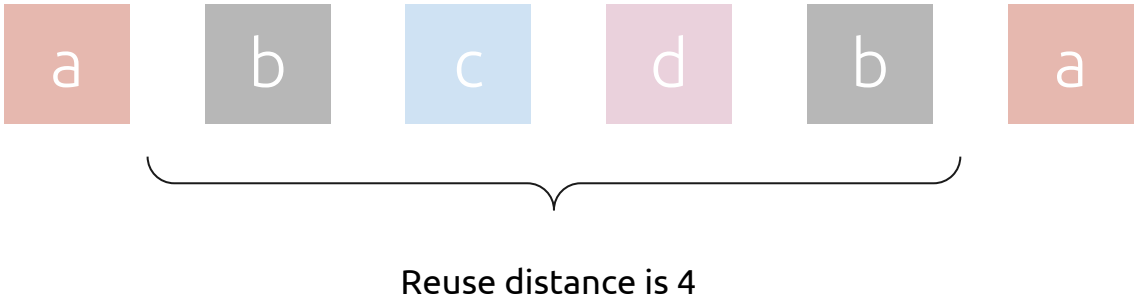


Figure 2: Example reuse distance

ReuseTracker - Integration Problems

- Originally developed top on **HPCToolKit** which handles all sampling and configuration.
- HPCToolKit is a **large dependency** → **aim** is to integrate **without HPCToolKit** which is basically **reproducing original paper**.
- **Progress**
  - ☒ PMU Sampling to detect uses.
  - ☒ Debug register configuration.
  - ☐ Reuse distance detection. (Under development)
  - ☒ On-time monitoring configuration. (**Problem**: Monitoring script starts a bit late than monitored script which resulted in loss of use samples)

Data Collection Tools - ComDetective

- ComDetective samples effective address involved in a memory access event that exceeds a threshold like 10000 loads(i.e 10000th load event triggers a sample)
- The thread that receives a sample halts execution, and looks up a global data structure where threads post their sampled addresses and thread IDs to see if another thread posted the same address
- If another thread published the same address, there is an inter-thread communication

Data Collection Tools - ComDetective

- Another way of detecting communication is using debug registers:
- When a thread receives a sample, it arms its debug registers with half of the cache line that the sampled address belongs to(explained in Debug Registers part)
- If a thread experiences a trap, it means that an address posted by another thread to the global structure was accessed by the thread and thus there was a communication
- The detected communications are represented and output in a matrix form to help analyze an applications inter-thread communication frequency

Data Collection Tools - ComDetective

- **Reconstruction of ComDetective**
- The ComDetective uses HPC Tool Kit to configure and monitor previously mentioned hardware units
- The kit comes with significant overhead and most of its utilutilitiesites are existent on the SuperTwin so ComDetective was integrated without using the kit
- While ComDetective hardcodes the program the monitor into a profiling program, due to the nature and goal of SuperTwin, the process ID of the program to monitor was passed into the reconstructed profiling program

Supercomputer Monitoring Tool

- **SuperTwin**:
  - Collaborates digital twin description and data extraction tools.
  - Implemented as a class:
    - Scaling from one CPU to a node, from a node to a cluster → Individual digital twins
    - Can be used to gather data and execute commands on a remote system
  - A SuperTwin instance:
    - Generates DTD instance of the remote system
    - Configures data collection tools
    - Configures local time-series and digital twin description database
    - Generates monitoring dashboards
    - Performs anomaly detection

Supercomputer Monitoring Tool - Design

- Main structure of the SuperTwin is developed by following:
  - Python
  - InfluxDB
  - MongoDB
  - Grafana Dashboards

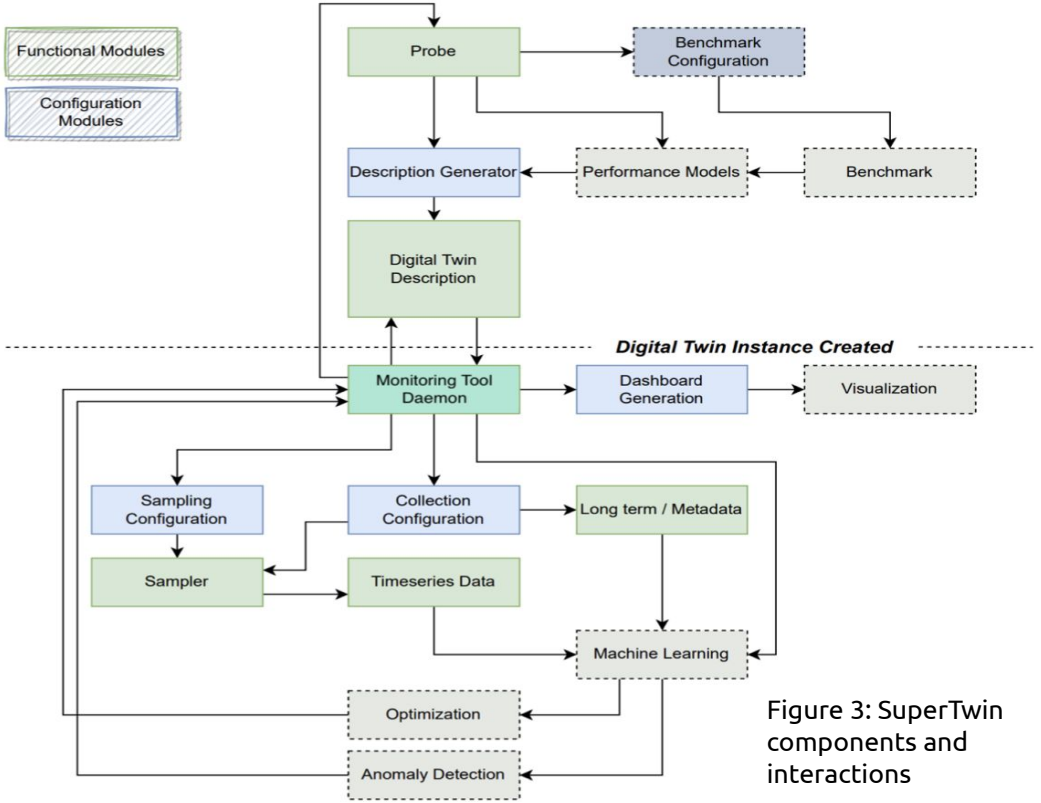


Figure 3: SuperTwin components and interactions

Supercomputer Monitoring Tool - Execution Flow

- SuperTwin enables monitoring of a supercomputer by executing following steps:
  - SuperTwin opens an SSH connection to the remote computing unit and configures PMDAs and PMCDs
  - Execute STREAM and HPCG benchmarks to profile the computing unit
  - Recursively generates DTD and copies it to the local MongoDB database
  - Configures local InfluxDB database to collect monitoring data from PMDAs
  - Generates Grafana dashboards by using DTD and queries InfluxDB database to display real-time monitoring data

Supercomputer Monitoring Tool - Scaling to Cluster

- Digital Twin is designed to monitor supercomputer clusters
- Need to scale the DTD generation and monitoring from a node to a cluster:
  - Recursively generate SuperTwin of each node
  - Configure head node PMDA to collect monitoring data from computing node PMDAs

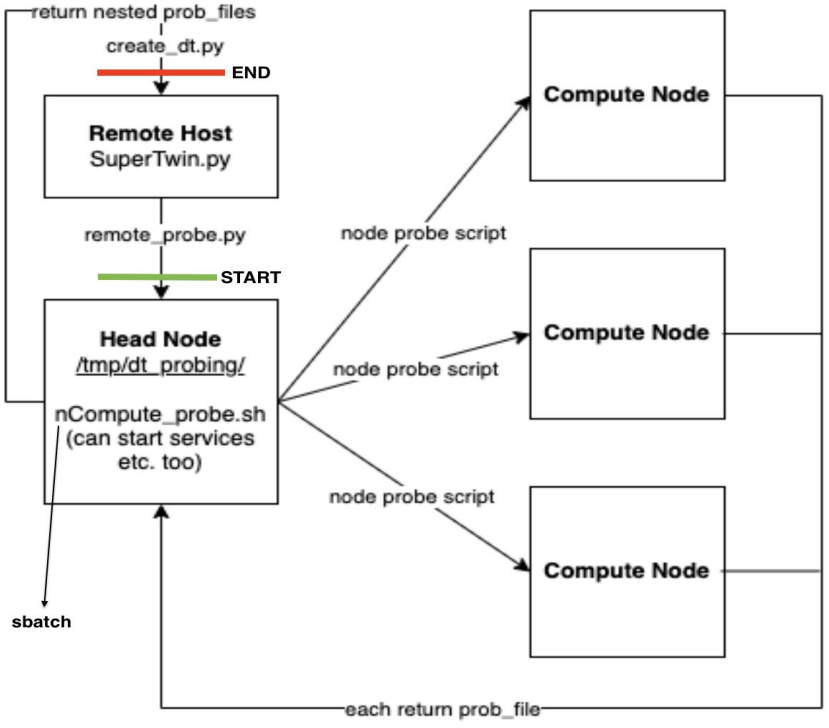


Figure 4: SuperTwin creation for clusters

Supercomputer Monitoring Tool - Scaling to Cluster

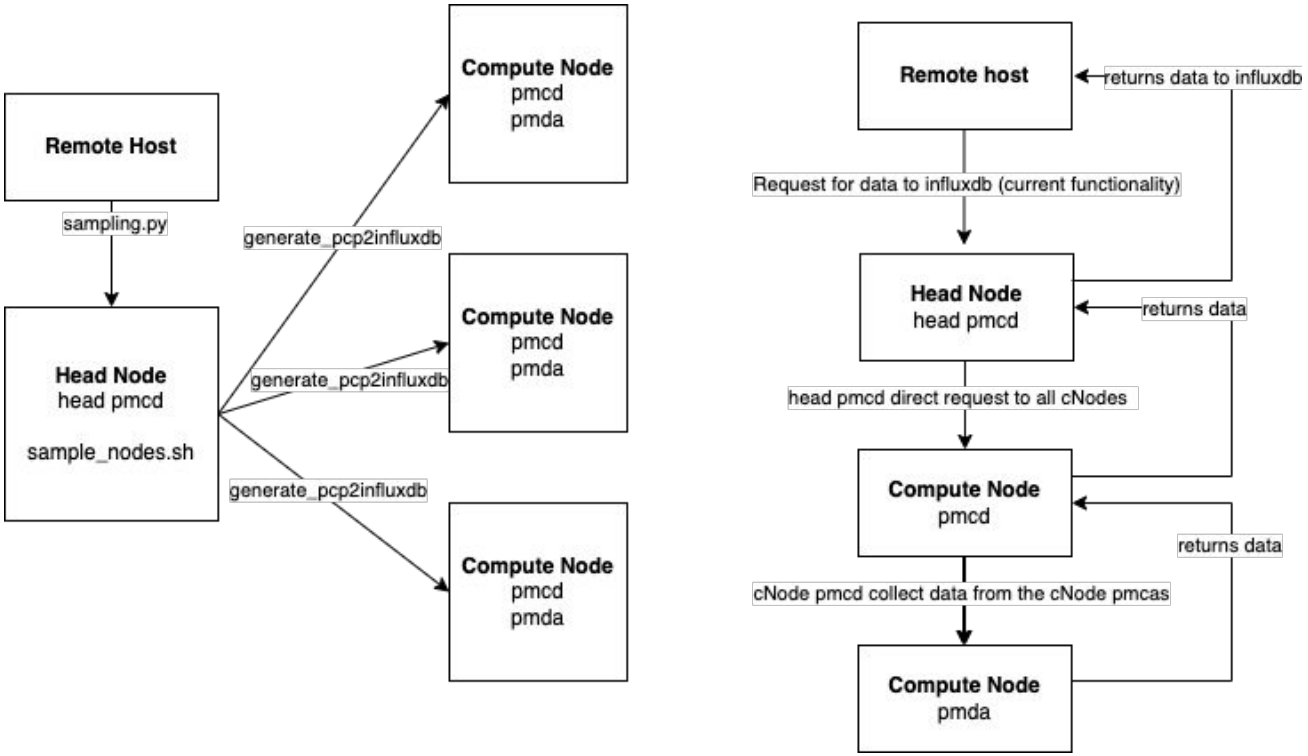


Figure 5: Cluster data collection with PCP

Supercomputer Monitoring Tool - Monitoring Dashboards

Figure 6: A monitoring dashboard

Integration, Deployment and Web Application

- Development process of the SuperTwin was complicated
- It has many dependencies → Libraries, specific OS tools etc.
- To manage these:
  - A CI/CD pipeline is implemented for testing and deployment
  - Tool is containerized with Docker → Can be used with different OSs
- SuperTwin can be used as a console application → Also have a web application to make it easy to use → initializing, selecting monitoring metrics, dashboard list, executing experiments
- Web application is implemented by using Flask and React

Integration, Deployment and Web Application

```
Remote probing is done..
Creating a new digital twin with id: 2bb28f62-f99e-4603-932b-aa4385ed19cb
Collection id: 63838642938892d328a60734
pcp2influxdb configuration: pcp_dolap_monitor.conf generated
A daemon with pid: 941 is started monitoring dolap
STREAM Benchmark thread set: [1, 2, 4, 8, 16, 22, 32, 44, 64, 88]
STREAM benchmark script generated..
STREAM benchmark result added to Digital Twin
Using database 'dolap_main' and tags 'tag=_monitor'.
Sending 35 metrics to InfluxDB at http://host.docker.internal:8086 every 1.0 sec...
(Ctrl-C to stop)
HPCG Benchmark thread set: [1, 2, 4, 8, 16, 22, 32, 44, 64, 88]
HPCG benchmark script, with params nx: 104 ny: 104 nz: 104 time: 60 is generated..
HPCG benchmark result added to Digital Twin
CARM config generated..
ADCARM Benchmark thread set: [1, 2, 4, 8, 16, 22, 32, 44, 64, 88]
adCARM benchmark script generated..
CARM benchmark result added to Digital Twin
Twin state is registered to db..
172.17.0.1 - - [27/Nov/2022 15:46:17] "POST /api/startSuperTwin HTTP/1.1" 200 -
172.17.0.1 - - [27/Nov/2022 15:46:22] "GET /api/setDB HTTP/1.1" 200 -
172.17.0.1 - - [27/Nov/2022 15:46:24] "GET /api/getMetrics/monitoring HTTP/1.1" 200 -
```

Figure 7: Web app - Backend

Integration, Deployment and Web Application

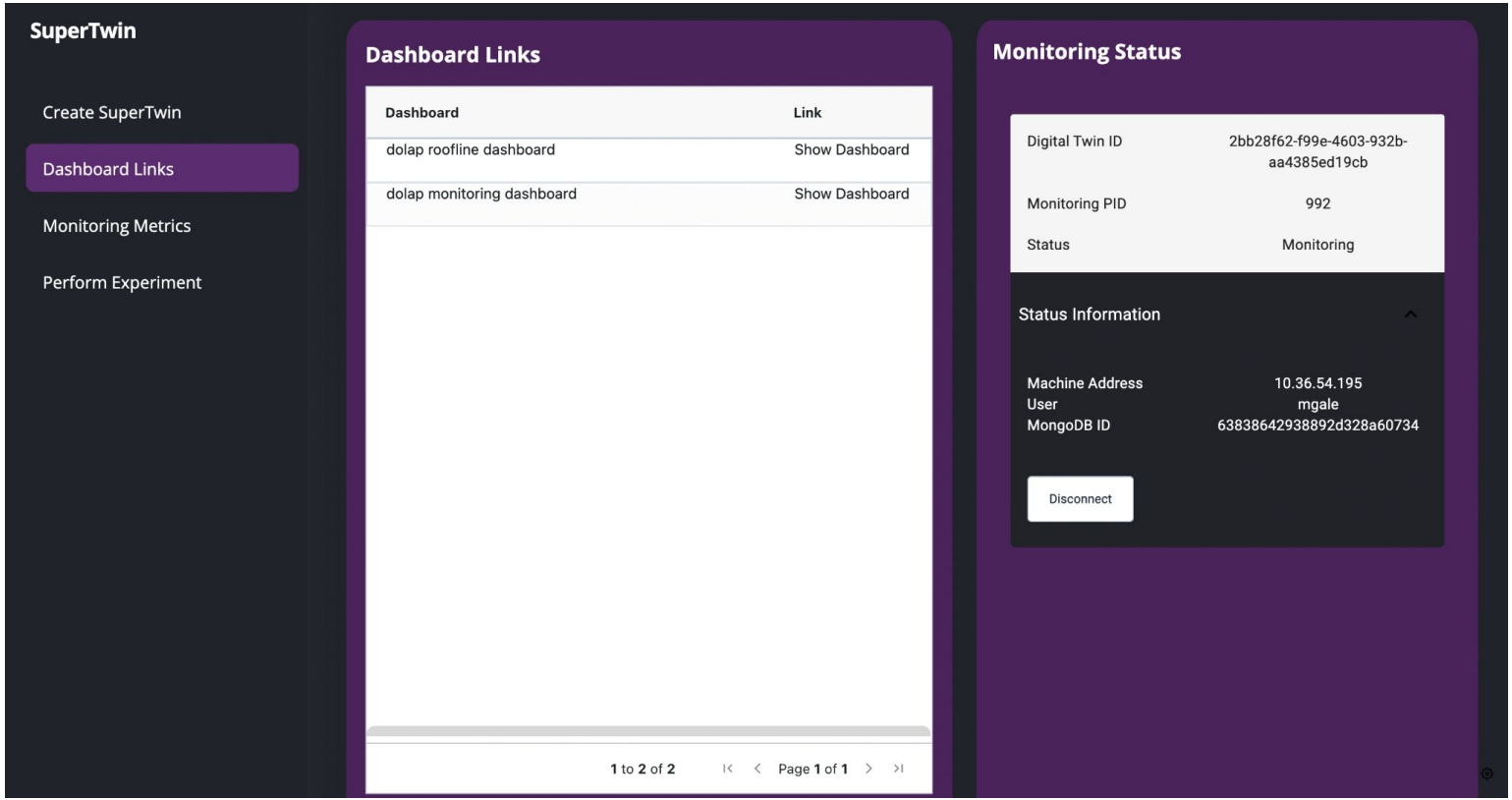


Figure 8: Web app - front end

Results

- SuperTwin, which allows real-time monitoring of supercomputers', is implemented. Also, a DTDL for Supercomputers is defined
- Console and web applications of the SuperTwin is deployed
- DTD and data collection strategies for cluster scaling with PCP are defined → Further scaling will be achieved by developing wrapper classes
- More metrics used compared to latest monitoring tools
- Extra two significant data collection tools is under development → ReuseTracker & ComDetective

Discussion

- Machine Learning interface to detect anomalies in super computer is not developed. Manual analysis of hundreds of metric is hard / not sustainable. Needs automatic anomaly detection mechanism
- Integration of Reuse Tracker and ComDetective is problematic. HPCToolKit is a crucial dependency.
- ReuseTracker and ComDetective uses debug register at the same time. Number of total registers is not enough to run both at the same time.