Centralized blacklisting system for PanDA

Fernando H. Barreiro Megino

Simone Campana

Alessandro di Girolamo

Jaroslava Schovancova

CERN IT-ES

Overview

1. Motivation

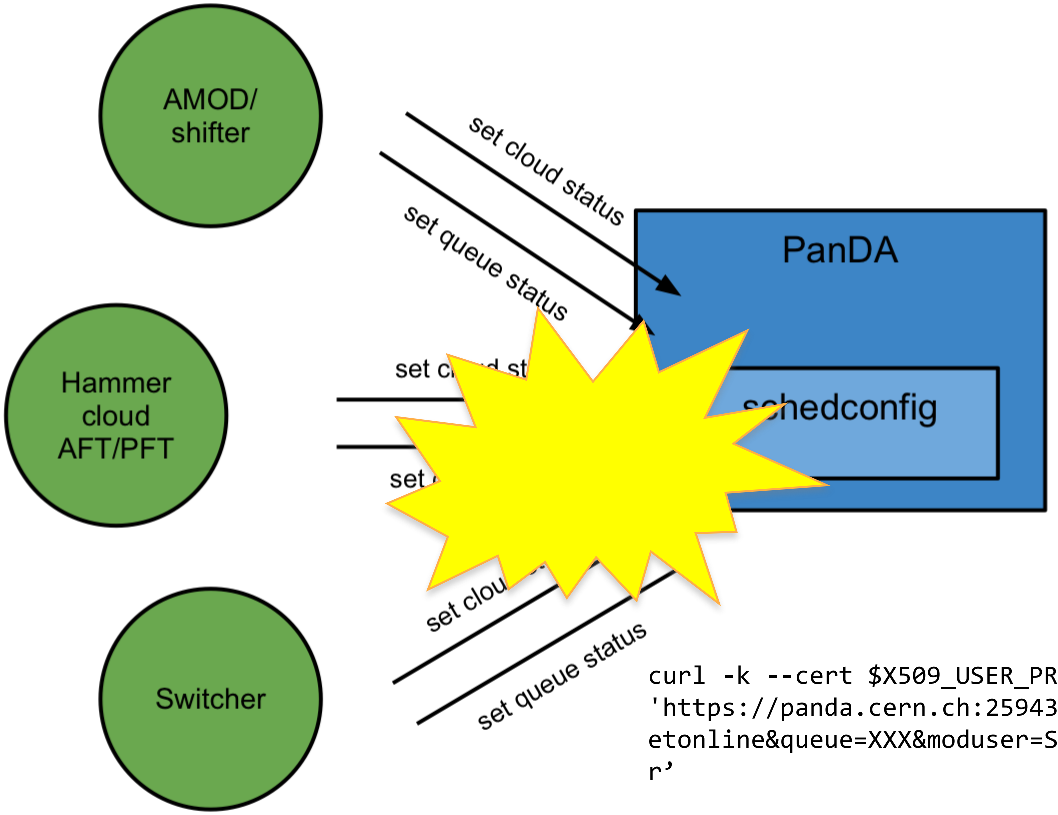
2. Overview of the proposed solution

1. Details of the system

2. Implementation steps

This presentation is meant as a guideline for discussion: Please interrupt for questions and comments

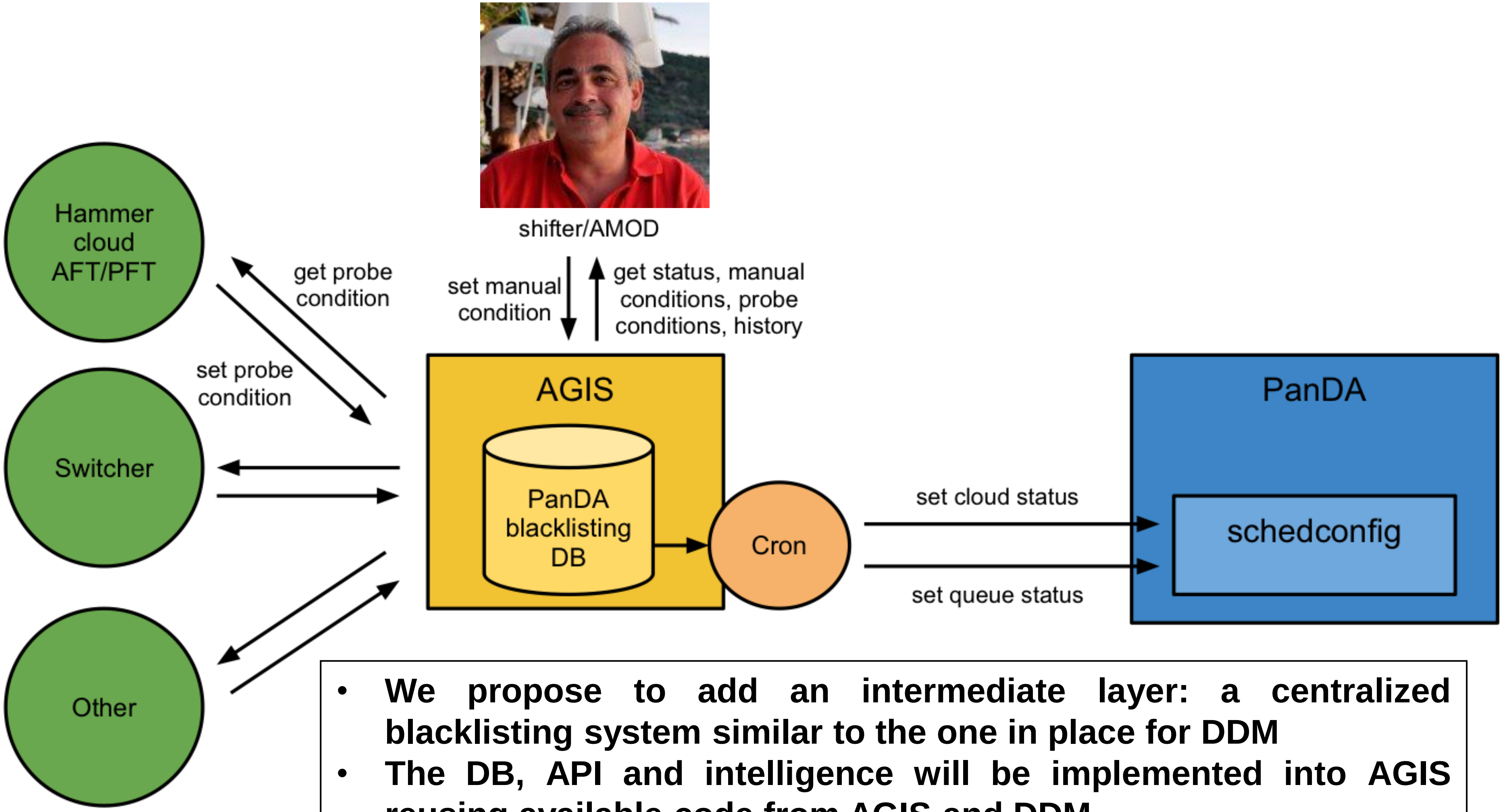
The current model



```
curl -k --cert $X509_USER_PROXY
'https://panda.cern.ch:25943/server/controller/query?tpmes=s
etonline&queue=XXX&moduser=ShifterName&comment=ELOG.TheNumbe
r’
```

- The current model works well in general
- Occasionally there are synchronization issues between the different agents setting the status
- PanDA does not know (and should not need to know) how to combine the statuses coming from different agents

The proposed model



- We propose to add an intermediate layer: a centralized blacklisting system similar to the one in place for DDM
- The DB, API and intelligence will be implemented into AGIS reusing available code from AGIS and DDM

Basic requirement

- Manage the status of **a queue or a cloud[1] for a given activity[2]**

1. PanDA currently is able to handle independently queue and cloud status. This reduces our work and should stay the same. The intermediate blacklisting layer will hold independent tables for clouds and will delegate on PanDA the combination of both.

2. Right now there are no activities. In the future we might be interested to manage independently the status of production and analysis

Conditions

- Conditions can be set at queue or cloud level and are of two types

1. **Manual conditions**
    - Responsibility of the operator (AMOD/shifter) to set the value
    - Values: **OFFLINE, TEST, BROKEROFF, ONLINE, <u>AUTO</u>**

1. **Probe conditions**
    - Written by the automatic agents
    - Values: **OFFLINE, TEST, BROKEROFF, ONLINE**
    - **No** AUTO probe condition
    - Current probes for queues: *Switcher* and *Hammercloud*
    - No current probes for clouds, but maybe they appear in the future
    - Hammercloud and Switcher would have to be adapted
        - instead of updating PanDA directly, they would have to update the new blacklisting system

Calculating the final status

- Conditions have to be combined to calculate the final status. Rules:
  1. Manual conditions overwrite probe conditions
  2. When manual condition is AUTO, the status will be set to the most restricting condition across the probes
     - Order of restriction: OFFLINE > TEST > BROKEROFF > ONLINE

MANUAL	PROBE 1	PROBE 2	STATUS
OFFLINE	*	*	**OFFLINE**
TEST	*	*	**TEST**
BROKEROFF	*	*	**BROKEROFF**
ONLINE	*	*	**ONLINE**
AUTO	OFFLINE	*	**OFFLINE**
AUTO	TEST	TEST | BROKEROFF | ONLINE	**TEST**
AUTO	BROKEROFF	BROKEROFF | ONLINE	**BROKEROFF**
AUTO	ONLINE	ONLINE	**ONLINE**

Other information

- Information stored with each condition update
  - **Comment:** The comment passed to schedconfig will be the first one available in the order: manual, switcher, hammercloud
  - **Timestamp**
  - **IP&DN of the submitter**: Only people with production role (or some other restriction) should be allowed to change the values of a condition
  - **Expiration date** of a condition (if appliable)

- The system will keep the history of changes for the manual and probe conditions

CLI/API

- Basic methods to handle queues and clouds
  - get/set manual condition
  - get manual condition history
  - get/set probe condition
  - get probe condition history
  - get status

- Any other call you miss from your AMOD experience?

Updating schedconfig

- The final status of the site needs to be periodically pushed/pulled to *schedconfig*

- There are **NO** changes needed in PanDA
  - Internal PanDA logic will be preserved
  - Existing PanDA commands into schedconfig will be used

- Initially a daemon would update schedconfig e.g. every 5 minutes
- Other models could be considered, such as schedconfig pulling the information or such as triggering updates immediately when the situation of a queue/cloud changes

Implementation steps

1. Identify a developer for this task

1. Add necessary blacklisting tables into AGIS schema
   - The tables will be strongly based on those implemented in the DDM centralized blacklisting system

2. Implement server side methods to interact with the DB

3. Implement client CLI&API
   - The CLI&API should be based on examples from AGIS

4. Modify Hammercloud and Switcher
   - Need integration/verification phase: HC and Switcher write simultaneously into schedconfig and the blacklisting system to verify the API, DB and logic work