PAUL SCHERRER INSTITUT

brightness

Michele Brambilla, Dominik Werder, Nikhil Biyani

ECP and File Writer control

BrightnESS WP5 Second Integration meeting
ELETTRA, Trieste

PAUL SCHERRER INSTITUT

FileWriter integration with NICOS

- new command format
- start & stop time
- status report & heartbeat
- initial values
- file/StreamMaster tagging

Command Format

- change NeXus structure ➠ **streams** in NeXus structure
- write **attributes**
- added **stop** vs **exit** command

```
{
  "nexus_structure": {
    "children": [
      {
        "children": [
          {
            "type": "stream",
            "stream": {
              "topic": "topic.with.multiple.sources",
              "source": "for_example_motor",
              "writer_module": "f142",
              "type": "float",
              "array_size": 4
            }
          },…
```

Command Format

- change NeXus structure ➠ **streams** in NeXus structure
- write **attributes**
- added **stop** vs **exit** command

Command Format

- change NeXus structure ➠ **streams** in NeXus structure
- write **attributes**
- added **stop** vs **exit** command

```
{
      "cmd": "FileWriter_stop",
      "job_id": "1234567890abcdef",
}
```

```
{
      "cmd": "FileWriter_exit"
}
```

Start and Stop

- start & stop **as soon as** the the command arrives

- provide a **specific time**:
  - start back in time ➠ RdKafka::offsetsForTimes + drop messages
  - start in future time ➠ drop messages
  - stop in future time ➠ FlatBuffer timestamp > StopTime + **Delta**
  - stop in past time ➠ same as before, but can be tricky

PAUL SCHERRER INSTITUT

Start and Stop

- start & stop **as soon as** the the command arrives

- provide a **specific time**:
  - start back in time ➠ RdKafka::offsetsForTimes + drop messages
  - start in future time ➠ drop messages
  - stop in future time ➠ FlatBuffer timestamp > StopTime + **Delta**
  - stop in past time ➠ same as before, but can be tricky

```
{
    "cmd": "FileWriter_new",
    "start_time" : 1234,
    "stop_time" : 12345,
    …
}
```

Start and Stop

- start & stop **as soon as** the the command arrives

- provide a **specific time**:
  - start back in time ➠ RdKafka::offsetsForTimes + drop messages
  - start in future time ➠ drop messages
  - stop in future time ➠ FlatBuffer timestamp > StopTime + **Delta**
  - stop in past time ➠ same as before, but can be tricky

```
{
    "cmd": "FileWriter_stop",
    "job_id": "1234567890abcdef",
    "stop_time": 1517242388000
}
```

PAUL SCHERRER INSTITUT PSI

Start and Stop

- start & stop **as soon as** the the command arrives

- provide a **specific time**:
  - start back in time ➠ RdKafka::offsetsForTimes + drop messages
  - start in future time ➠ drop messages
  - stop in future time ➠ FlatBuffer timestamp > StopTime + **Delta**
  - stop in past time ➠ same as before, but can be tricky

- **assumptions**:
  - start & stop time in **milliseconds** (in command)
  - FlatBuffer **pulse_time** (not kafka timestamp) in milliseconds

- **works** with NICOS

PAUL SCHERRER INSTITUT

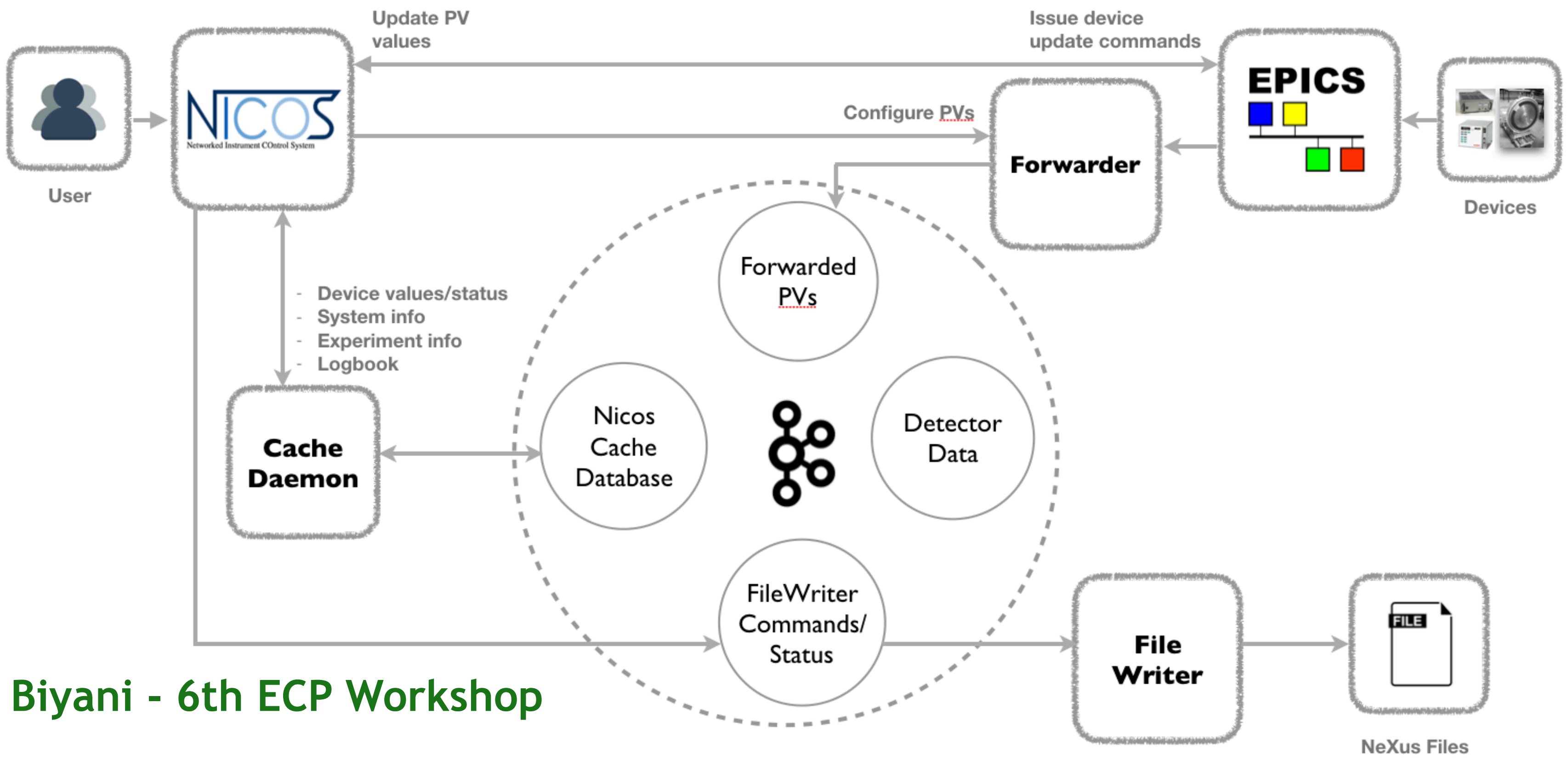
Start & Stop: possible issues (and solution)

- start & stop **as soon as** the the command arrives
- provide a **specific time**:
  - start back in time ➠ RdKafka::offsetsForTimes + drop messages
  - start in future time ➠ drop messages
  - stop in future time ➠ FlatBuffer timestamp > StopTime + **Delta**
  - stop in past time ➠ same as before, but can be tricky
- there are **many sources** in a topic ➠ keep a **sources count**, eventually **decrease** the counter; when there are **no sources close the stream**
- one source can **stop producing messages** (for any reason) **before** StopTime is reached ➠ if msg->err() == **End Of Partition** && **system time** > StopTime + Delta remove source
  - solves slow sources: **Delta** > time between messages in slower source

Initial values

- A source in a topic could not have received updates since long time. How much back in the log has to file writer to dig?

  1. we can force the forward to "**refresh**" PVs every x minutes
  2. NICOS already has a **cache**, we can use the "last known value"



Initial values

- A source in a topic could not have received updates since long time. How much back in the log has to file writer to dig?

    1. we can force the forward to "**refresh**" PVs every x minutes
    2. NICOS already has a **cache**, we can use the "last known value"

- Pros and cons

    1. more traffic on the network, more work for the forwader
    2. due to latencies on the network the "last known value" NICOS knows could not be the actual last value

- Combine the two? NICOS proposes a value, if there's no more recent in the forwarder accept NICOS one

Status report & heartbeat

- each StreamMaster (*i.e.* file) regularly provides a report of the consumed **messages** (number and bytes), **errors** and **status** (running, finished, error status)

- the report says when the next report is **expected** - works as an **heartbeat**

- in addition the report contains information for **each stream:** bandwidth, message frequency, message size

```
{
    "type": "stream_master_status",
    "next_message_eta_ms": 2000,
    "stream_master": {
        "state": "running",
        "messages": 100.0,
        "Mbytes": 1000.0,
        "errors": 0.0,
        "runtime": 22002
    },
    "streamer": {
        ...
    }
}
```

Status report & heartbeat

- could be integrated with the file writer (Master) info:
  - **advantages**:
    one single message; no need to figure out which file produced it;
  - **disadvantages**:
    can’t figure out which file produced it; a failure in the stream could affect the master process; components are disentangled ➠ simplicity;

```
{
  "type": "filewriter_status_master",
  "files": {
    "0000000000000004": {
      "filename": "a-dummy-name.h5",
      "topics": {}
    }
  }
}
```

Extra

- **configuration** file is fully working, added new options
  - can configure default **output path**
  - can be used to configure file writer & **kafka** options
- logging: added **severity levels** in the form of `enum class Sev`
  - only in file writer (no forwarder)
- each file is tagged using a **unique id** (specified in start command)
  - the unique id has to be given to **stop** each write process
  - the report from the stream can be **associated** with the **file**

Ansible management & other remarks

- currently each repository (file writer, forwarder, amor simulation) has its own **ansible** folder
- on the ansible "**master**" machine there is a **copy** of the tools **source**
- on the ansible "**target**" machine there is a **copy** of the **ansible** scripts
- we can get rid of this using a "dmsc-ansible" **repository** and removing ansible folders form single packages
  - easier to maintain: dependencies can be uniquely defined, avoid multiple installation same dependency, …
- **dm-dev-env** repo is outdated: shall we use a new machine (see Nikhil **dm-sinq-amor**) or remove the repository?
- who is in charge to handle "**file is already open**" ? NICOS or file writer?
  - **NICOS "proposes"** the file name; eventually the **file writer append** some id (timestamp?); NICOS gets the file name from the file writer status