lbrulibs – Grafana Integration

Will Panduro Vazquez

ND UD Meeting
12/03/2021

Build & Test Status

- Standalone tests with daq_application back up and running
  - Recent changes to infrastructure (post 2.8.2) meant config JSON had to be refreshed to match latest from new readoutmodules package
  - Works with nightly from Nov 23rd – potentially needs further tweaks for newer versions to match updates to queue configuration
  - Readout team confirm that the configuration should now be stable, so will updated and continue tracking in the run-up to 2.10

Grafana Integration

- Key Components
  - Publication of monitored quantities from lbrulibs via opmonlib interface
  - Availability of InfluxDB to store published information
    - Can create local instance in container via Pocket
    - Centrally managed facility available in NP04 (used for this test)
      - Need to request login to np04-serv systems (np04-onl-admins@cern.ch)
      - Note – to access git from here need to set up proxy
        - In .gitconfig, add:

```
[http]
        proxy = http://np04-web-proxy.cern.ch:3128
        sslVerify = false
```

  - Configuration of Grafana dashboard, reading from InfluxDB
    - Need to be invited to register with Grafana instance
      - For me this came from Giovanna, not sure if there is an ‘official’ person to ask

Opmonlib

- Configuration based on .jsonnet schema files (as per config)
  - Currently pacmancardreaderinfo.jsonnet
- Quantities to be monitored should be specified here, along with type
- Insert command into CMakeLists file to generate C++ headers based on schema

sourcecode > lbrulibs > schema > lbrulibs > pacmancardreaderinfo.jsonnet

```
// This is the application info schema used by the data link handler module.
// It describes the information object structure passed by the application
// for operational monitoring

local moo = import "moo.jsonnet";
local s = moo.oschema.schema("dunedaq.lbrulibs.pacmancardreaderinfo");

local info = {
    uint8  : s.number("uint8", "u8",
                      doc="An unsigned of 8 bytes"),
    float8 : s.number("float8", "f8",
                      doc="A float of 8 bytes"),
    choice : s.boolean("Choice"),
    string : s.string("String", moo.re.ident,
                      doc="A string field"),

   info: s.record("ZMQLinkInfo", [
       s.field("num_packets_received",          self.uint8,     0, doc="Number of packets received"),

   ], doc="ZMQ Link Information"),
};

moo.oschema.sort_select(info)
```

sourcecode > lbrulibs > CMakeLists.txt

```
cmake_minimum_required(VERSION 3.12)
project(lbrulibs VERSION 1.0.5)

find_package(daq-cmake REQUIRED)

daq_setup_environment()

find_package(appfwk REQUIRED)
find_package(logging REQUIRED)
find_package(ers REQUIRED)
find_package(detdataformats REQUIRED)
find_package(readoutlibs REQUIRED)
find_package(ndreadoutlibs REQUIRED)
find_package(ipm REQUIRED)
find_package(Boost COMPONENTS unit_test_framework REQUIRED)

daq_codegen(pacmancardreader.jsonnet TEMPLATES Structs.hpp.j2 Nlis.hpp.j2 )
daq_codegen(pacmancardreaderinfo.jsonnet DEP_PKGS opmonlib TEMPLATES opmonlib/InfoStructs.hpp.j2 opmonlib/InfoNlis.hpp.j2 )
```

Opmonlib

- Within C++ source, include generated headers
- Module must also implement ‘get_info’ method, which is called by monitoring system periodically to query values of quantities of interest
  - Passed info container, which can be routed to different levels of code to fill in desired values

sourcecode > lbrulibs > plugins > PacmanCardReader.hpp > ...

```
/**
 * @file PacmanCardReader.hpp PACMAN card reader DAQ Module.
 *
 * This is part of the DUNE DAQ , copyright 2021.
 * Licensing/copyright details are in the COPYING file that you should have
 * received with this code.
 */
#ifndef LBRULIBS_PLUGINS_PACMANCARDREADER_HPP_
#define LBRULIBS_PLUGINS_PACMANCARDREADER_HPP_

#include "appfwk/cmd/Structs.hpp"
#include "appfwk/cmd/Nljs.hpp"
#include "appfwk/app/Nljs.hpp"

#include "lbrulibs/pacmancardreader/Nljs.hpp"
#include "lbrulibs/pacmancardreaderinfo/InfoNljs.hpp"
```

```
void PacmanCardReader::get_info(opmonlib::InfoCollector& ci, int level){

  m_zmqlink[0]->get_info(ci, level);

}
```

```
virtual void get_info(opmonlib::InfoCollector& ci, int /*level*/){
  dunedaq::lbrulibs::pacmancardreaderinfo::ZMQLinkInfo linkInfo;

  linkInfo.num_packets_received = m_packetCounter;

  ci.add(linkInfo);
}
```

```
namespace dunedaq::lbrulibs {

class PacmanCardReader : public dunedaq::appfwk::DAQModule
{
public:
  /**
   * @brief PacmanCardReader Constructor
   * @param name Instance name for this PacmanCardReader instance
   */
  explicit PacmanCardReader(const std::string& name);

  PacmanCardReader(const PacmanCardReader&) =
    delete; ///< PacmanCardReader is not copy-constructible
  PacmanCardReader& operator=(const PacmanCardReader&) =
    delete; ///< PacmanCardReader is not copy-assignable
  PacmanCardReader(PacmanCardReader&&) =
    delete; ///< PacmanCardReader is not move-constructible
  PacmanCardReader& operator=(PacmanCardReader&&) =
    delete; ///< PacmanCardReader is not move-assignable

  void init(const data_t& args) override;
  void get_info(opmonlib::InfoCollector& ci, int level) override;
```

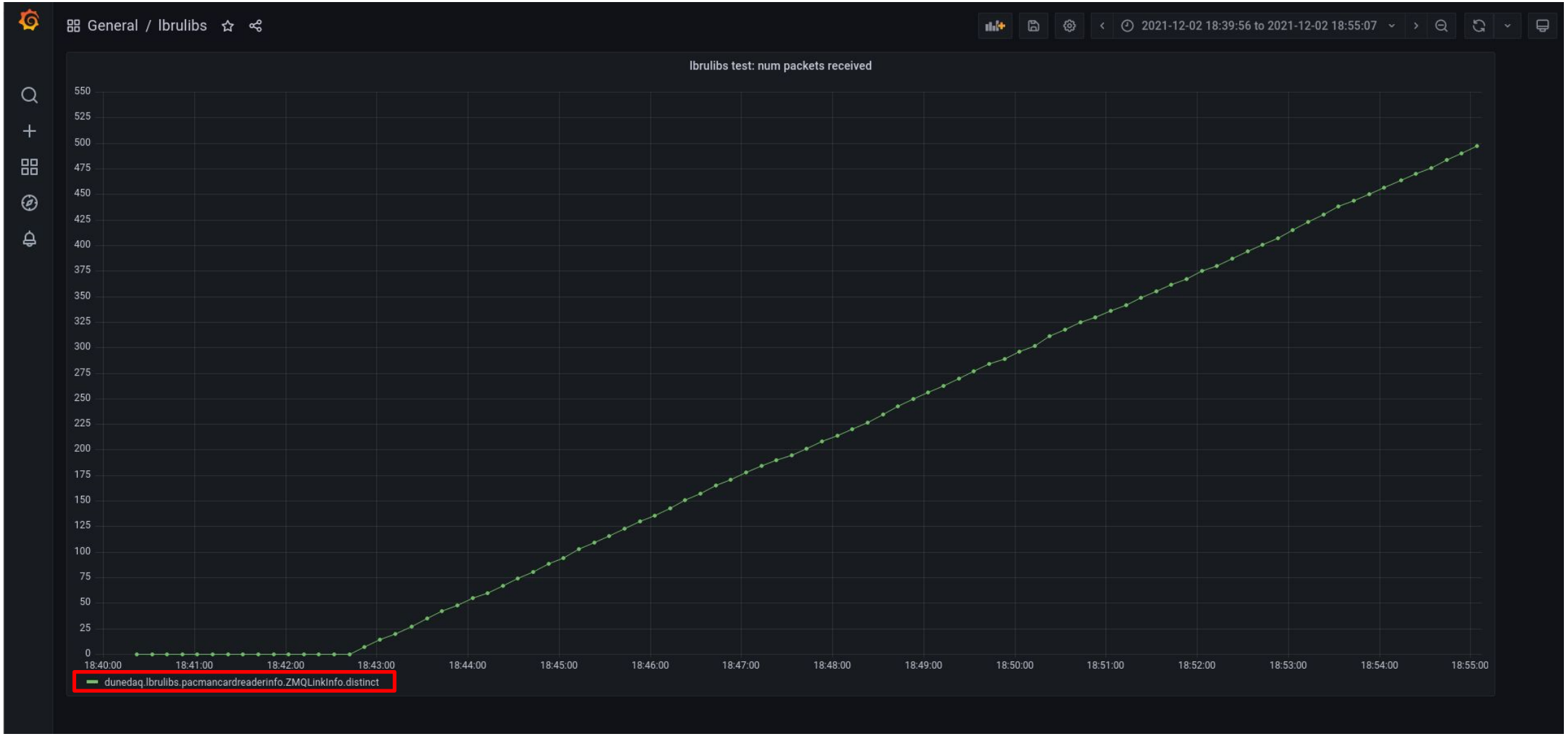
Testing in NP04

- Recommended to run on np04-srv-012 (most suitable for DAQ)
- Invoke daq_application as usual, but add parameter to invoke write to influx
  - Using Krzysztof's python packet generator as data source

```
[jpanduro@np04-srv-012 ~]$ daq_application -n pacman -c fake_NDreadout.json -i influx://188.185.88.195:80/write?db=db1
```

- Should then automatically create structures within influxDB and write data, to be picked up in Grafana
- In Grafana, create new dashboard and navigate to data source
  - Dashboards can then be saved for future testing

Results!


General / lbrulibs
2021-12-02 18:39:56 to 2021-12-02 18:55:07
lbrulibs test: num packets received
550
525
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
18:40:00
18:41:00
18:42:00
18:43:00
18:44:00
18:45:00
18:46:00
18:47:00
18:48:00
18:49:00
18:50:00
18:51:00
18:52:00
18:53:00
18:54:00
18:55:00
dunedaq.lbrulibs.pacmancardreaderinfo.ZMQLinkInfo.distinct

Next Steps

- Review existing unit tests using opmon based on changes needed to get it working with influx & commit updated code
- Discuss what else we want to monitor and publish!