Realtime display of EPICS data using HTML5 and Websockets

September 2016 EPICS Meeting

Jonathan Hanks

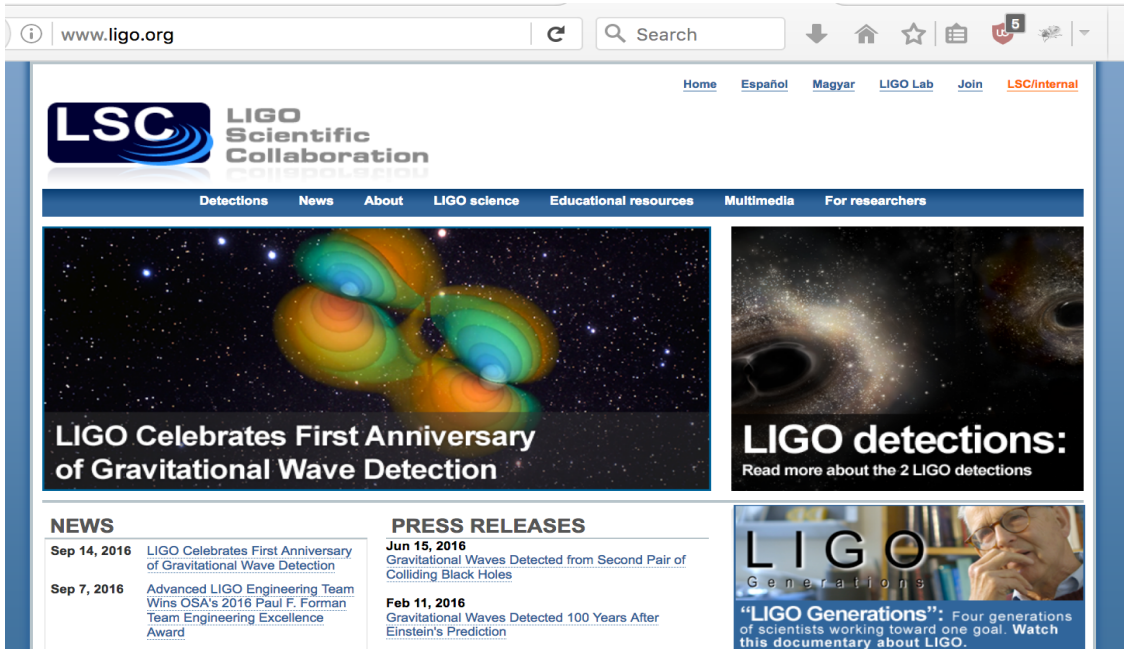
LIGO Laboratory

California Institute of Technology

Forr

The State of LIGO

LIGO had two confirmed detections during the first data taking run with the new instrument!



For details see: http://papers.ligo.org

LIGO

Current Status

- Currently we are working to improve the LIGO interferometer
  - » Higher power laser
  - » More sensitive instrument
- Start the next data taking run Q4 '16

Using EPICS in LIGO

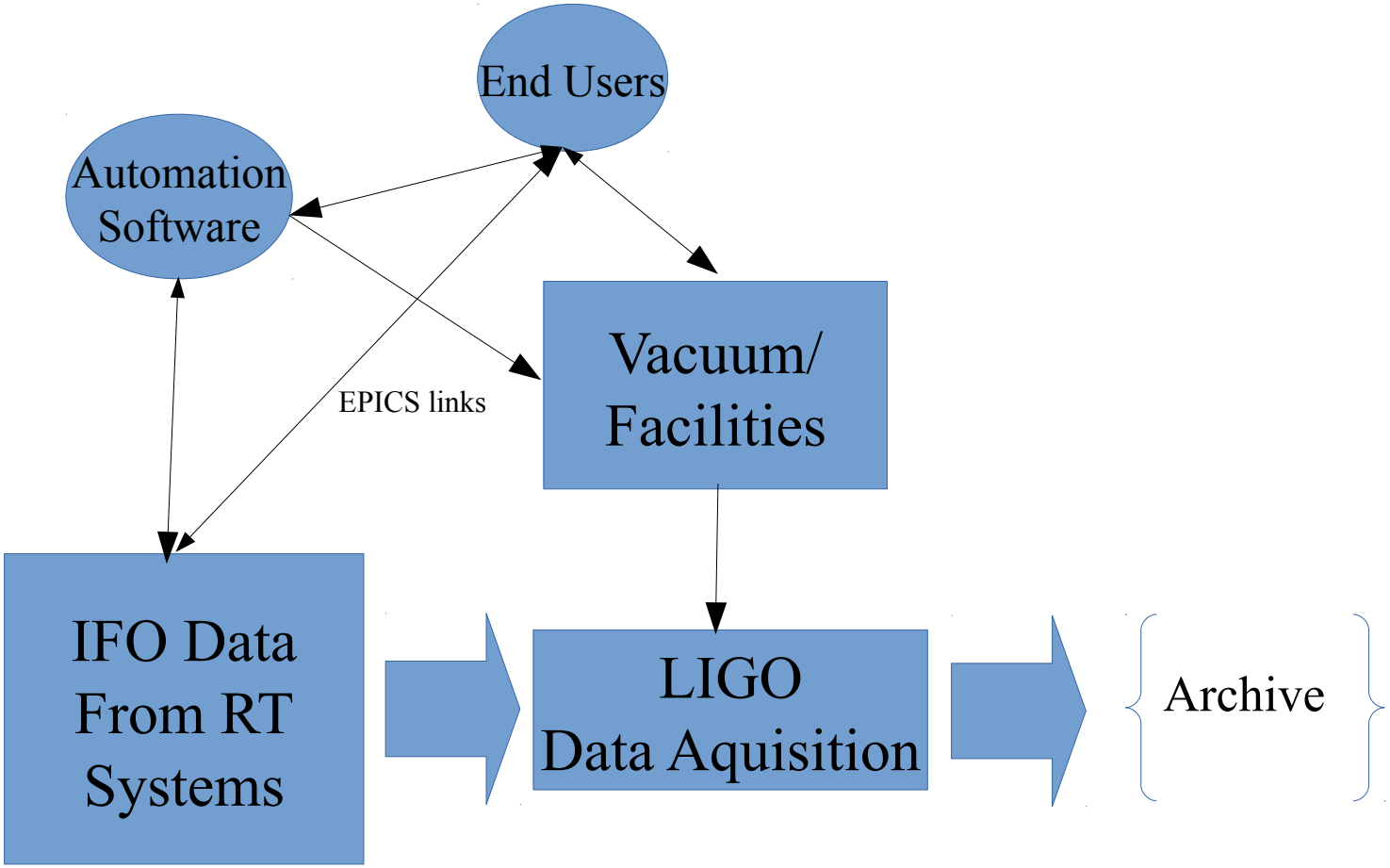
- EPICS is how our controls automation system interacts with the LIGO interferometers.
- The human/computer interface
  - » Our system uses a custom RT Linux solution on x86
  - » Phasing out VxWorks systems for Beckhoff systems
- The main data stream is through an in-house system
  - » Our ‘slow’ systems, vacuum, environment are run on the Beckhoff systems (and some legacy VxWorks systems we are retiring).
  - » For these systems both control and data acquisition take place through EPICS

LIGO

Scale of our EPICS use

- Our main automation/control system drives the IFOs via EPICS
- Most control room use is done via EPICS
- RO gateway allows remote collaborators/commissioners access
  - » +1000 collaborating researchers, engineers, …
  - » RO access with collaboration credentials required.

How EPICS Connects LIGO



Tooling/Remote Access

- We are based on PyEPICS, MEDM, StripTool, and some custom tools (EPICS CA scripting).
  - » Looking at moving off of MEDM/StripTool before they stop working with new versions of EPICS Base.
- Remote Access
  - » Modified MEDM to pull screens over http.
  - » This has been extremely useful for our remote collaborators.
  - » Targeting OS X and Linux with this effort.
  - » Most successful on OS X with an internal mac-ports repository.
  - » Linux packaging ongoing.

Supporting our Tooling

While our remote MEDM has been very successful I do not want to be supporting all the LIGO remote users desktops/devices.

So ….

LIGO

Realizations

- Everyone has a browser, even on their phone.
- Looked at existing web based solutions
  - » Webopi v3.x – too much overhead
  - » Ajax requests fave too much overhead and do not update enough.
- Web Sockets this give us a pretty good connection into the browser.

LIGO

Why not build it?

- A websocket based CA gateway should
  - Be fairly easy to do
  - Lightweight

So, why not build something.

This started as a side project on my time, there is still a lot to do. But even with the small amount of time that I have put into it, works well.

In testing at Hanford now

- Server side, custom solution (hybrid Go/C++)
  - C++ - Gives us EPICS CA
  - Go – Very easy to build a scalable highly concurrent network services, including good websocket support.
- Python based adl to HTML5/SVG/JS
  - Adl → intermediate → html5/SVG/JS
- Client side
  - Webopi – used for the wire protocol and existing JS
  - Knockout.js – dynamic interface

LIGO

Server

Stand alone application

- No external runtime
- Just need the EPICS shared libs, libc, libzmq
  - » And a shared lib that is part of the application.
- Though I do proxy it behind apache so that I have more flexible authentication options.

Small

- ~1300 lines of Go (not counting the websocket library)
- ~400 lines of C++

Highly concurrent and easy to reason about. Message passing and queuing instead of mutexes and condition variables.

The client JS understands R/W, but the server just drops all write commands

LIGO

Translation Layer

Two staged approach.

1) .adl → intermediate

2) Intermediate → dynamic HTML/SVG/JS

By using an intermediate layer the MEDM parser doesn't need to know about the final JS logic and we are better able to decouple sections.

Still need to implement more translation. Much is working, high fidelity layout of many elements, groupings, … but more to do.

Intermediate layer

Specify this on the intermediate

<input pv="H0:VAC-LX_X1_PT140B_PRESS_TORR" pv_output="exponential" style="color: #ffffff; background-color: #000000; width: 80px" type="text"/>

<rect pv="H0:VAC-MY_GV12_II239_AIP_IC_LOGMA" pv_attr="no_text,dyn_geometry" y="522+51*(1-pv/100)" height="51*(pv/100)" width="15" x="887"></rect>

Then translate to the slightly more complex final version (using webpda and nockout.js).

<input data-bind="value: _pvs[5]().value().toExponential(_pvs[5]().prec()),css: { _disconnected: _pvs[5]().connected() == false }" pv="H0:VAC-LX_X1_PT140B_PRESS_TORR" pv_output="exponential" style="color: #ffffff; background-color: #000000; width: 80px" type="text"/>

<rect data-bind="attr: { y: 522 + 51*(1-_pvs[60]().value()/100), height: 51*(_pvs[60]().value()/100) },css: { _disconnected: _pvs[60]().connected() == false }" pv="H0:VAC-MY_GV12_II239_AIP_IC_LOGMA" pv_attr="no_text,dyn_geometry" width="15" x="887"></rect>

LIGO

Reports/Status

- Users like it.
- They can get their information via phones, tablets, and computers.
- I like not supporting custom software on users systems.

LIGO

Future

- Where ever LIGO goes (CSS/QtCaDM/…) I want to have browser based access to our screens and tools.
  - » We will support the display managers in the control room and test stands.
  - » But I really want to get out of the business of supporting display manager install and runtime issues.

Pie in the Sky

- It shouldn't be too hard to make a full blown display manager from this.
  - The server is small and self contained.
  - Add a Qt interface with a web browser control…
  - Then who knows...

Image Creative Commons 3 (see https://commons.wikimedia.org/wiki/File:Pumpkin-Pie-Slice.jpg)

LIGO

Questions?