Using *gallery* for data access

Marc Paterno

LArSoft Usability Workshop

22 June 2016

New UPS package: *gallery*

- *art* 2.0 was released the week of May 16.
- A major feature was the separation of the event-processing framework code in *art* from the persisted data structure support, which was moved to a new UPS product, *canvas*.
- At the same time, we released the first version of *gallery*, which is a product that supports reading *art*/ROOT data files outside of the *art* framework executable.
- At the same time, LArSoft has introduced three new UPS products, containing the data products defined by LArSoft, and the data products in *nutools* were moved into *nusimdata*.
- The **distribution bundle** *larsoftobj* was introduced to give a single-command installation for all the UPS products needed to use *gallery* to read LArSoft-created *art*/ROOT files.

Installation

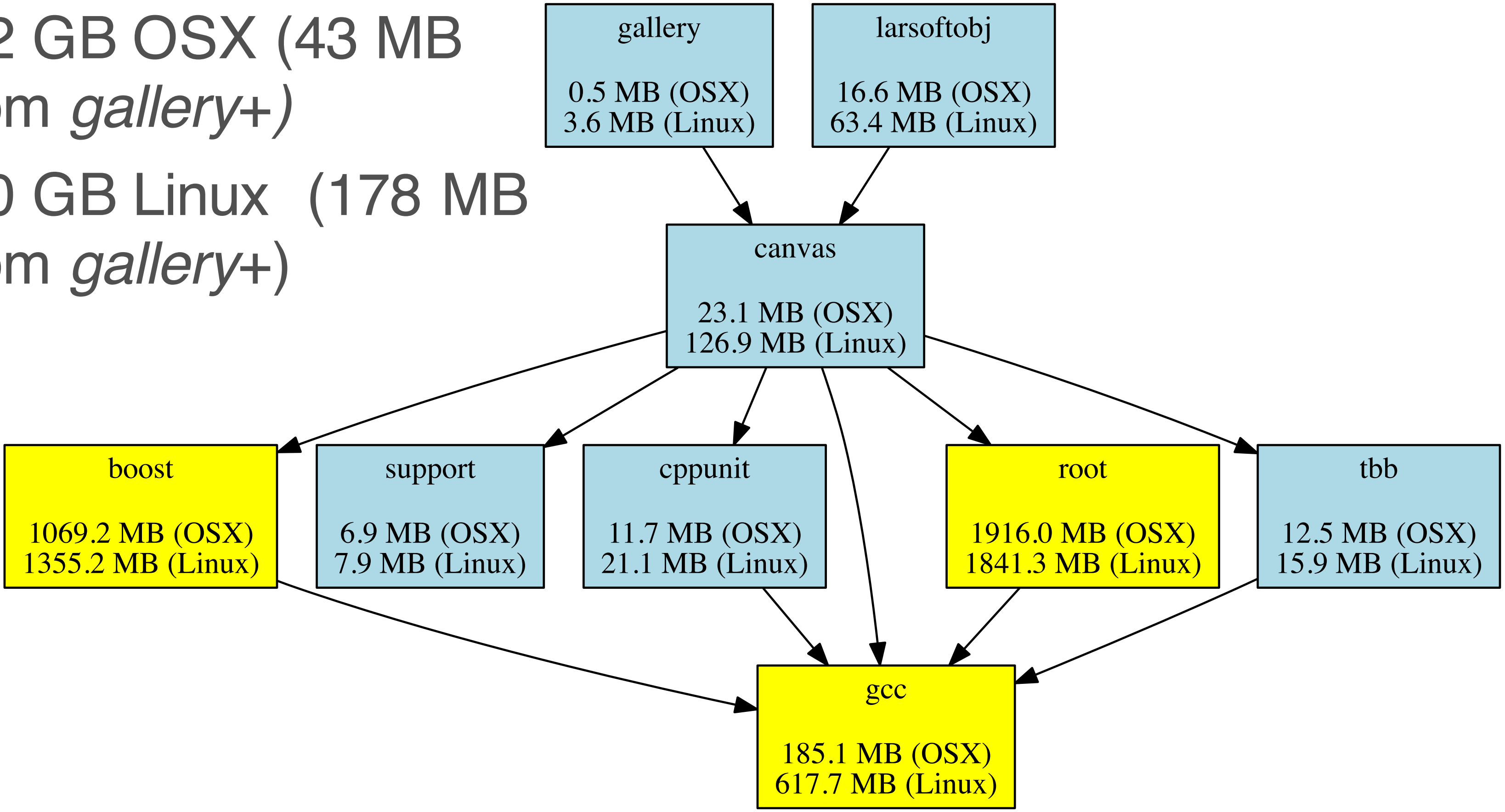
- As of the time of this presentation, installation is available for:
  - Yosemite
  - Ubuntu 14.04 LTS
  - SLF 6.x and 7.x (suitable for RHEL-based distributions, e.g. CentOS)
- Installation instructions are at http://scisoft.fnal.gov/scisoft/bundles/larsoftobj/ (look for the newest version, and view the HTML file for instructions)
- It is the usual “run pullProducts with the right arguments”.
- **Caveat**: PyROOT and ROOT macro support on Yosemite is limited by an incompatibility in ROOT’s LLVM version; the ROOT team are working on moving to a newer version of LLVM which is needed for fixing the problem.

What is *gallery* for?

- *gallery* provides access to event data in *art*/Root files outside the *art* event processing framework executable:
  - without the use of EDProducers, EDAnalyzers, *etc*., thus
  - without the facilities of the framework (*e.g.* callbacks from framework transitions, writing of *art*/ROOT files).
- You can use *gallery* to write:
  - compiled C++ programs,
  - ROOT macros,
  - Using PyROOT, Python scripts.
- You can invoke any code you want to compile against and link to.
  - Be careful to avoid introducing binary incompatibilities.

Installation (suitable for this demo)

- `./pullProducts <dir> <os> larsoftobj-v1_02_00 e10 prof`
- *os* can be slf6, slf7, d14, u14
- 3.2 GB OSX (43 MB from *gallery*+)
- 4.0 GB Linux (178 MB from *gallery*+)



Contributions welcome

- This is an early version of *gallery*: contributions (within the constraints of given above) are welcome.

Demonstration

- Using compiled C++
- Using a ROOT macro
- Using PyROOT

Please ask questions. Demos will be done on Ubuntu 14.04 and Yosemite, but everything works on SLF6, SLF7, and related RHEL 6&7 distributions.

Caveats and recommendations

- The compiled C++ program option is the most robust.
- The interactive ROOT macro usage allows the flexibility of interacting with ROOT objects.
  - A bug in ROOT can cause crashes when using ACLiC.
  - Until we have a fix from ROOT, avoid ACLiC here.
- PyROOT is the least robust.
  - Many failures on OSX due to an old LLVM version in ROOT.
  - Specific failures on Linux because of limitations in PyROOT's data model.
- My recommendation: use compiled C++ whenever possible, and interactive ROOT when you really want the interactivity. Use PyROOT only when you require the user of other Python libraries; be prepared to work around defects in the model.