uB DAQ and EPICS working together on Linux

Glenn Horton-Smith

2011/06/15

Overview

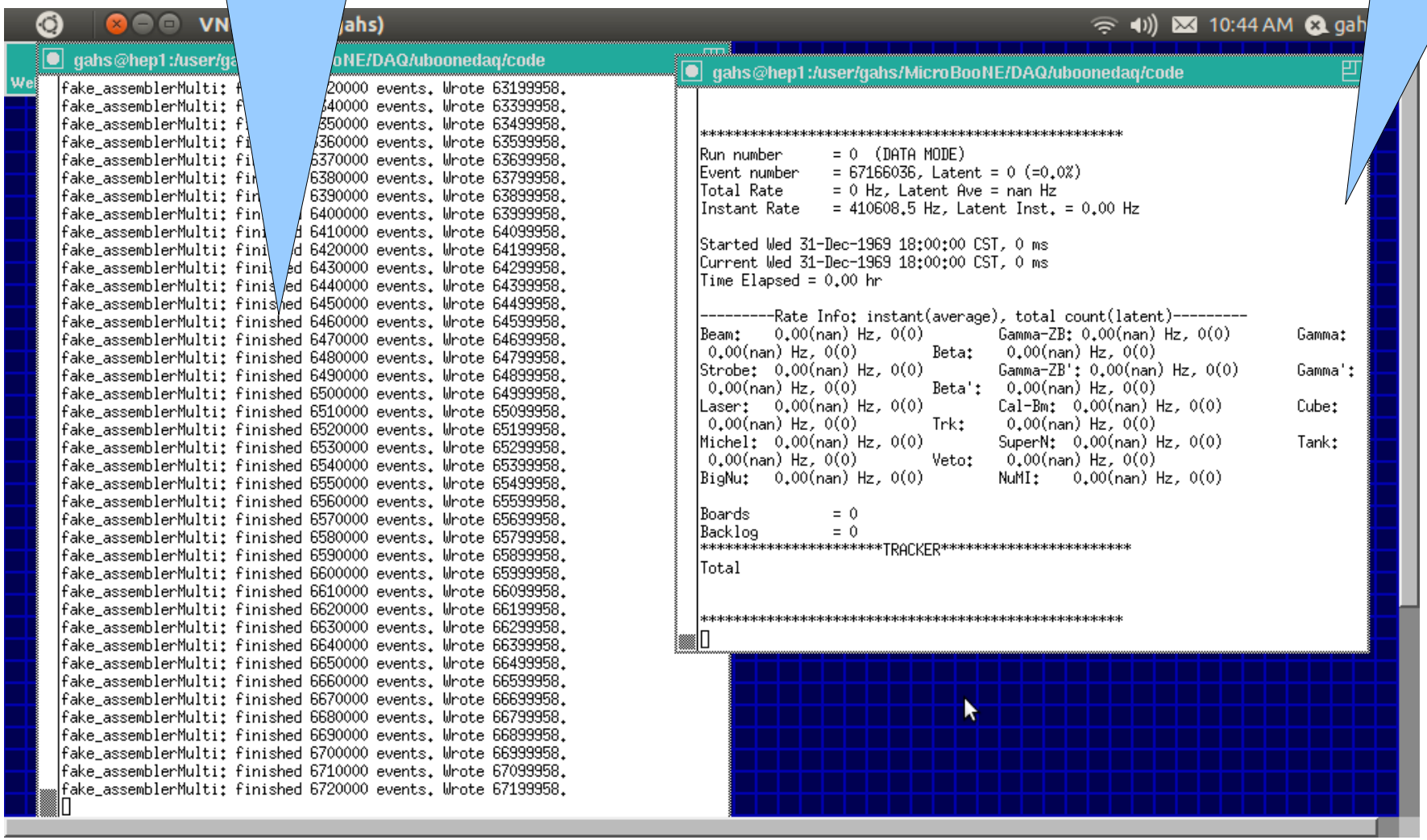
- “uB DAQ” running on Linux.
  - fake_seb, fake_assemblerMulti, run control display
  - Same performance with SEB and assembler on different hosts as when all on same host. (local GB/s ethernet)
  - shared memory for assembler-RC communication
- EPICS running on Linux
  - base: channels for readout and control; devices
  - extensions and modules: GUIs, sequencer, archivers
- Added device to EPICS base for uB DAQ RC "channels"
  - It wasn't actually that hard.
  - Proof of principle for EPICS-based run monitor, control, sequence, saved history.
- Next tasks and things to decide.

Mock uB DAQ running on Linux.

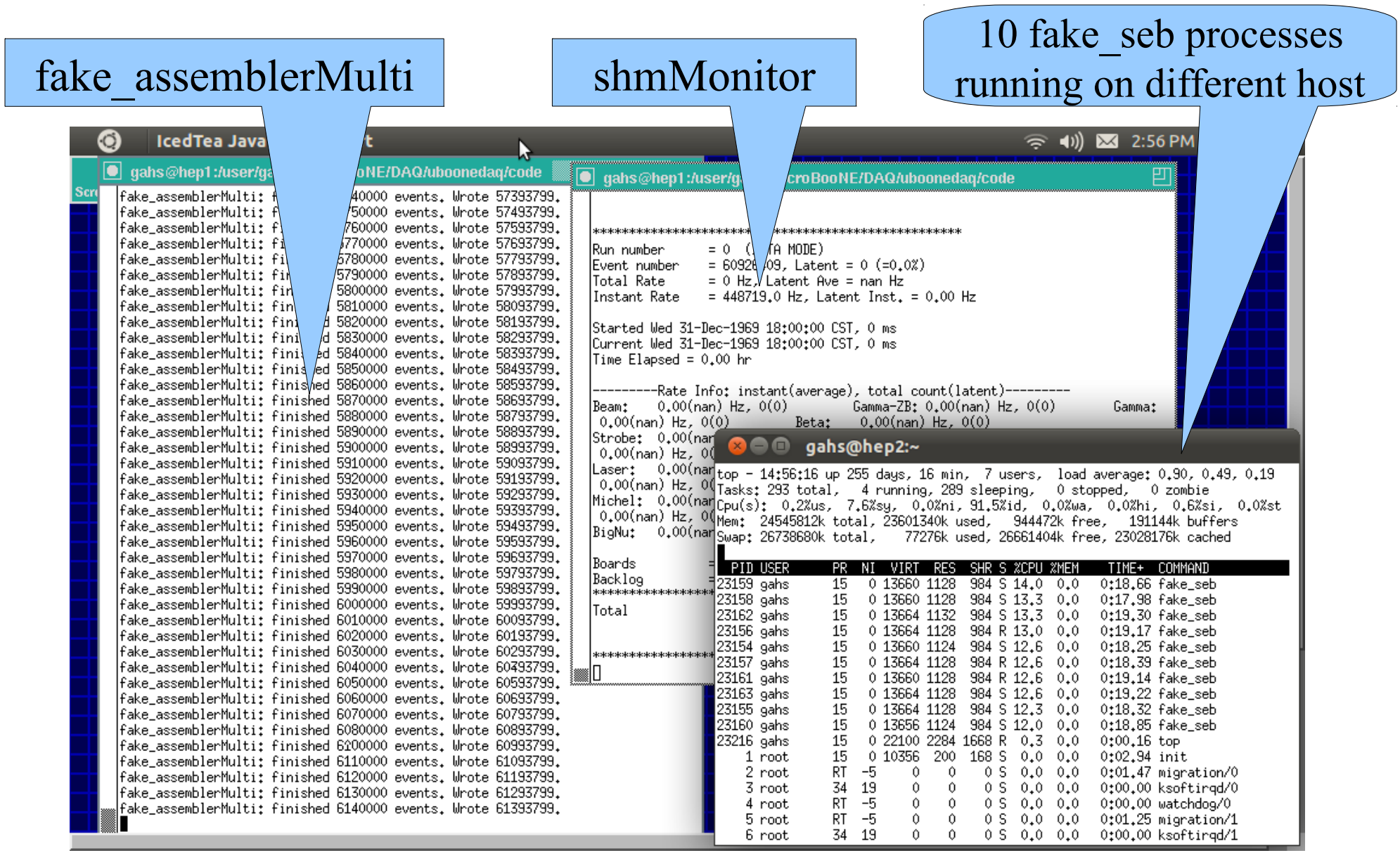
fake_assemblerMulti

10 fake_seb processes
also running on same host

shmMonitor



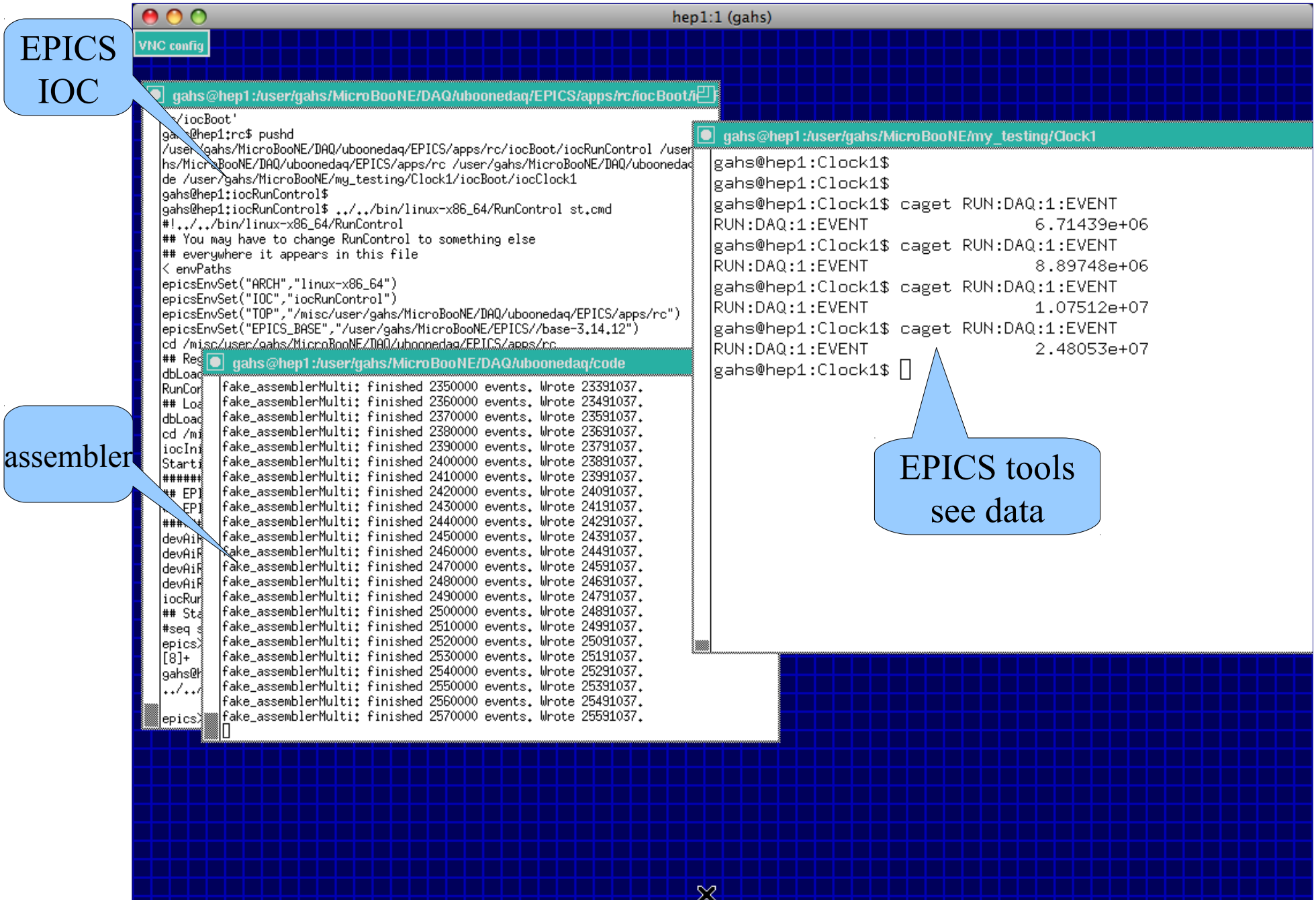
Same or higher performance with (mock) SEBs on one host and assembler on the other



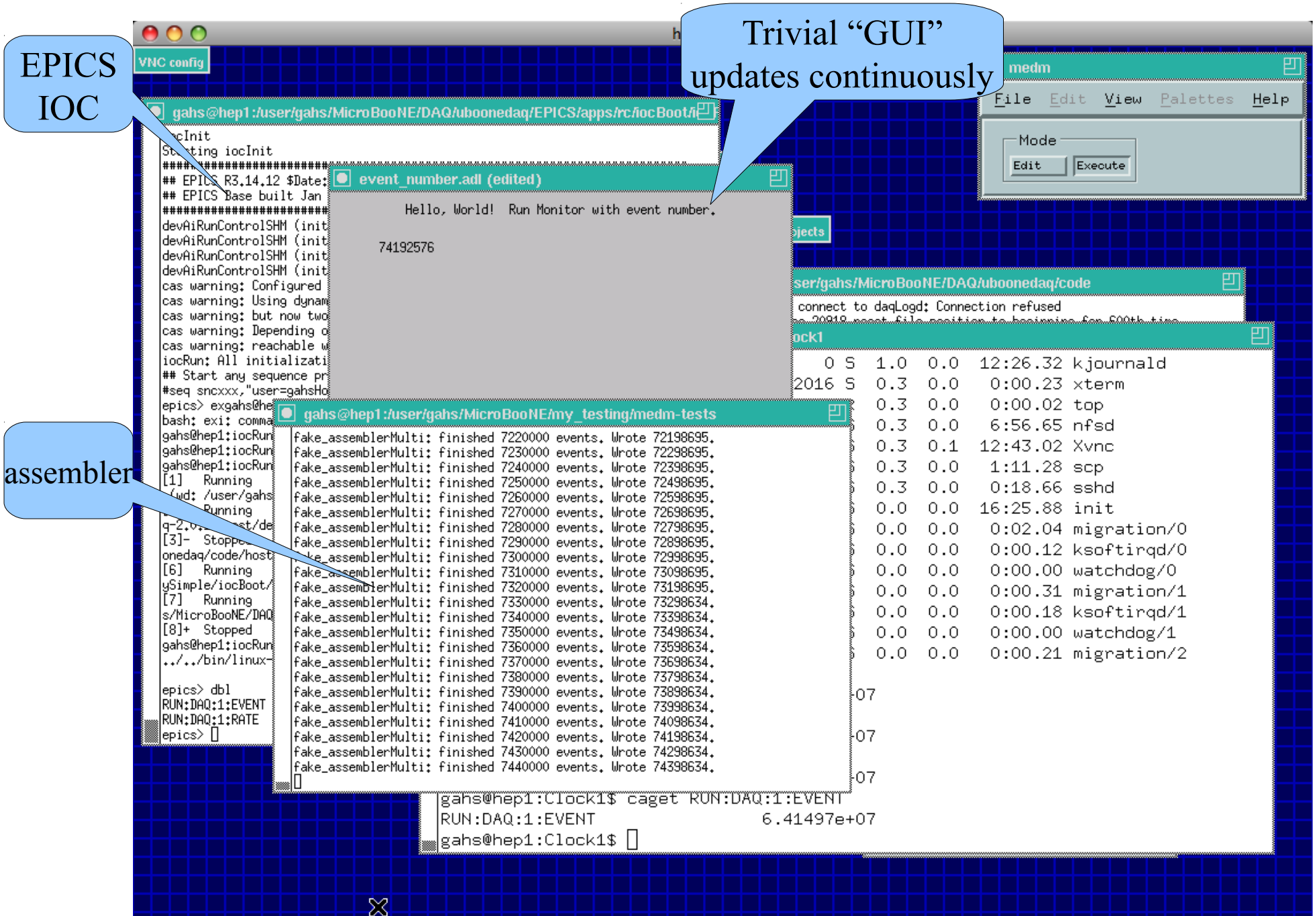
EPICS: components and status of my testing on Linux for MicroBooNE

Component	Source	Functionality	
Base (R3.14.12)	ANL	Process Variable, channel access backbone	I tested
Sncseq (2.0.12)	BESSY	State sequencer	I tested
Alarm Handler (1.2.26)	ANL	Alarm reporter and manager	I will test
Display and control panels: multiple options (just three shown below)			
EDM	ORNL	Motif-based, old	-
MEDM	APS	Motif-based, supported	I tested
CAML	ORNL	Browser-based, new	I will test
History / archiving: multiple options (just two shown)			
ChannelArchiver	was ORNL	file-based solution	-
RDBChannelArchiver	ORNL	SQL-based, new	I will test
Other: multiple Python and LabView interface options,			-

Separate EPICS IOC and uB assembler processes, shared memory



With a trivial MEDM display



Next things to try

- RDBChannelArchiver
- AlarmHandler
- Some more useful monitoring devices
  - I've written an IOC for monitoring a Linux host's state: CPU load, uptime, free memory, CPU temperature.
  - Would be neat to track history of this on uboonedaq01.
  - For CPU temperature, someone needs to configure the right sensors on uboonedaq01.fnal.gov.
- CAML display and control panels

Some things to decide

- Which EPICS components to choose for gui, history recording (“archiving”).
- What naming convention to adopt for EPICS channels.

EPICS user interfaces

- **Motif Editor and Display Manager** (APS) is well-known, easy. One display per window.
- **Channel Access Markup Language** (ORNL) is "web browser based" using plug-ins installed in browser, allowing navigation between pages.
- Other options include **edm, StripTool, DM2K,** and coding a gui ourselves (*e.g.*, in Python).
- I intend to look at CAML when I get a chance.

EPICS record naming conventions

- Geoff mentioned last meeting the desirability of having a convention from the start.
- For the “Run Control” example, I used a convention that could be summarized as

  `SPECIES:AREA:UNIT:PROPERTY`

- Examples:

  RUN:DAQ:1:EVENT, RUN:DAQ:1:RATE,
  HVPS:TPC:1:VOLT, HVPS:TPC:1:CURR,
  HVPS:PMT:1:VOLT, *etc*.

Elaborating on SPECIES:AREA:UNIT:PROPERTY

- SPECIES is “an alphanumeric primary name describing a general device class.”
  - “Examples are BPMO for beam position monitor, QUAD for quadrupole... and so forth.”
  - This is not the same as the EPICS type such as “analog input”.
- AREA indicates a region or subsystem.
- UNIT is a decimal digit number.
- PROPERTY is “an alphanumeric secondary name representing some attribute of the device.”

Some of the text above is taken from Chapter 1 of the SLC Control System Principles of Operation (March 21, 1994).

Better conventions might be devised, but ...

- whatever convention we adopt, it should be
  - easy to interpret for humans,
  - easy to remember (both the convention and the names used in the convention),
  - systematic enough that we can imagine writing tools to do things like scan the values ofr all records of a certain “species”, etc.
- The SPEC:AREA:UNIT:PROP convention seems to fit the bill, something like it has worked before.