External Control of JANA processes via janactl

David Lawrence JLab

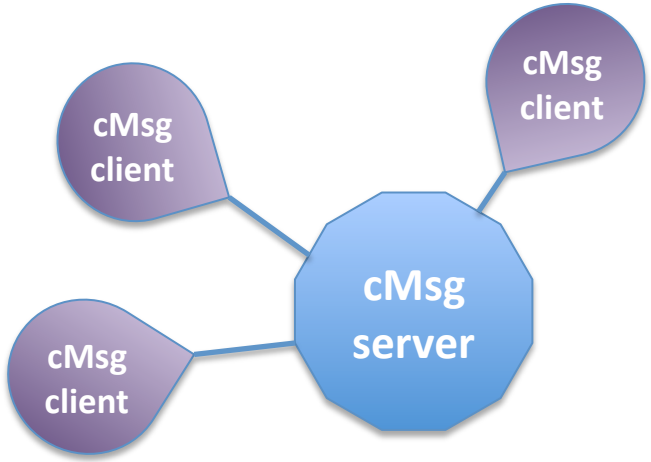
Jan. 6, 2010

JANA

- JANA is an event processing framework under which the Hall-D reconstruction code is being developed
  - Inter-algorithm data passing
  - Calibrations/conditions database access
  - Geometry access
  - Multi-threaded
- As the basis of the reconstruction code in Hall-D, JANA will be used:
  - on the monitoring farm
  - for the level3 trigger farm

cMsg system

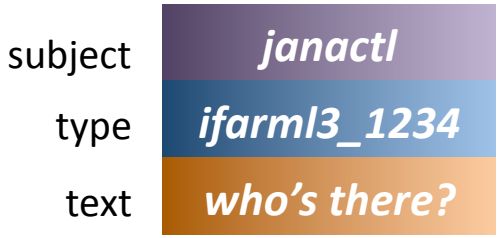
- cMsg is a publish/subscribe system that can pass messages over a network
  - Relatively simple API
  - Broadcast-like capability (many recipients)
  - Targeted (single recipient)
  - Supports multiple payloads of various data types keyed by string



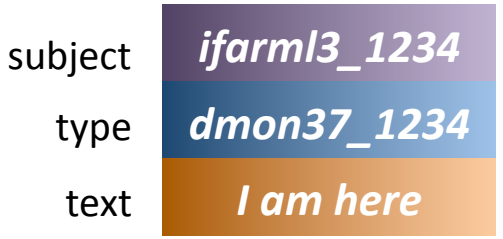
JANA control messages

- cMsg messages address recipients using a *subject* and a *type* (both free-form strings)
- janactl convention has all messages use *subject* only for addressing recipient(s) and the *type* to specify the sender
  - Allows recipient to easily respond directly to sender
- Control command is always passed in the *text* field of the message
  - Descriptive strings (easier to debug)
  - Additional data is attached to message as payloads

message sent to discover existing processes



message sent in reply



Currently Implemented Commands

Commands *janactl* plugin understands

- *"who's there?"*
- *"get threads"*
- *"pause"*
- *"resume"*
- *"quit"*
- *"kill"*

Replies *janactl* plugin sends

- *"I am here"*
- *"thread info"*

All communications are done using a passive model (i.e. *send and forget*).

cMsg supports *send and wait* messaging but a design decision was made to not use that feature.

Using *janactl*

- *janactl* is the name used for both the plugin and a command-line utility
- Any JANA program can attach the *janactl* plugin to allow external monitoring and control
  - Data Quality monitoring
  - Level 3 trigger
  - Offline farm jobs
  - Local jobs (e.g. processing simulated data)

Thread info. via *janactl*

This output shows the result of a "*get threads*" command issued when 2 processes were listening.

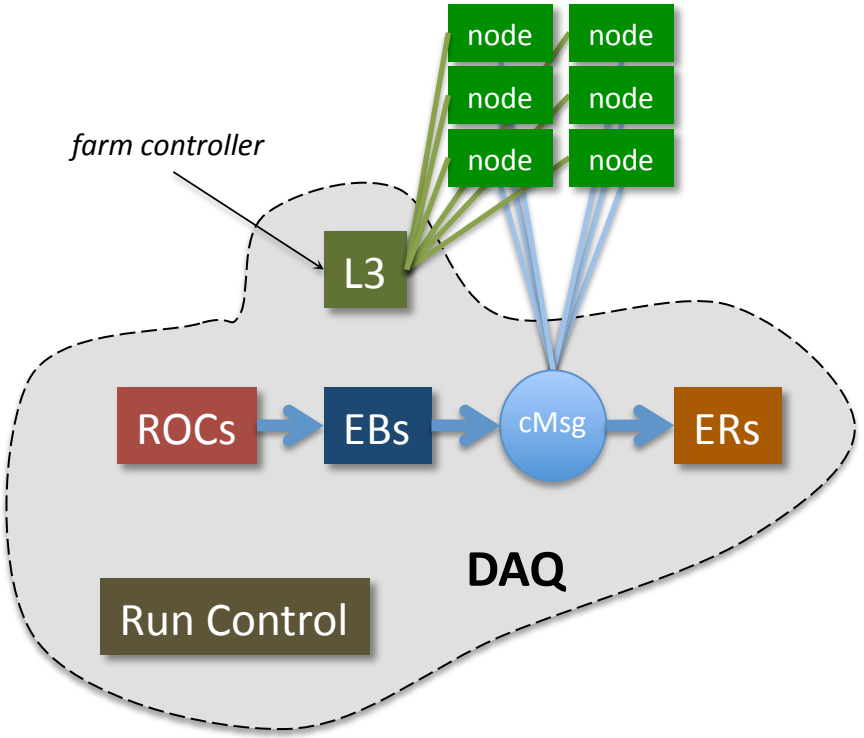
One had a single processing thread while the other had three processing threads

```
~>janactl thinfo -t 0.1
parseUDL: udl remainder = localhost/cMsg/janactl
parseUDL: host = localhost
parseUDL: host = Amelia.local
parseUDL: mustMulticast = 0
parseUDL: TCP port = 45000
parseUDL: subdomain = cMsg
parseUDL: subdomain remainder = janactl, len = 7
DONE PARSING UDL
Sent command: get threads to janactl

Threads by process:
-------------------------------------------------
Amelia.local_29219:
   thr. 0xb0491000  109 events  2.63513Hz (4.33184Hz avg.)
Amelia.local_29220:
   thr. 0xb0491000  42 events  3.29637Hz (1.98569Hz avg.)
   thr. 0xb0513000  42 events  3.32588Hz (2.00329Hz avg.)
   thr. 0xb0595000  33 events  1.63243Hz (1.57239Hz avg.)
```

```
Usage:
       janactl [options] cmd

Communicate with remote or local processes that have the janactl
plugin attached.

Options:

   -h, --help  Print this message
   -t timeout  Set the timeout of commands while waiting for a response.
   -u udl      Set UDL of cMsg server. If not given, the JANACTL_UDL environment
               variable is used. If that's not set, the localhost is used.
   -d descr.   Set description text of this program for use by cMsg server.
   -s subject  Send command to subject (default is all "janactl" which is
               all processes.

   --timeout timeout      Same as -t
   --udl udl              Same as -u
   --name name            Same as -n
   --description descr.   Same as -d
   --subject subject      Same as -s

The janactl system uses a passive comminucation mechansim for control. Specifically,
messages are sent, but responses (if any) are received asynchronously. As such
commands such as the "list" command simply broadcasts a query to all processes
that askes for them to send back a message notifying us of their existence. At
some point, we must decide all responses have been received and continue on.
the "timeout" for this is set by default to 3 seconds, but an alternate may be set
via the "-t timeout" command line switch. Commands that expect only a single
response will continue as soon as that response is recieved.

 commands:
   list        Lists available processes.
   pause       Pause remote process(es)
   resume      Resume remote process(es)
   quit        Quit remote process(es) gracefully
   kill        Kill remote process(es) harshly
   thinfo      Get thread info. for remote process(es)
```

Integration with DAQ system

- A L3 farm will be a component of the DAQ system
- The farm is a nebulous entity in that nodes/processes may come and go during normal operation
- The proposed model would use a *farm controller* program that would present the farm to the DAQ as a single *codaobject*



Final Words …

- Availability
  - *janactl* will be in JANA 0.6.0 which will be released by Jan. 15th
- Limitations
  - Command set is fairly limited
  - Passive communication model
    - Some error codes won't be available
    - Additional delays
    - Disappearing peer must be implied
- Future
  - Enhance command set
  - Write farm controller program using *janactl*