Run Control, Databases and Calibration

Murrough Landon – 1 March 2001

http://www.hep.ph.qmw.ac.uk/~landon/talks

Overview

- Crate Controller framework
- IS classes and IGUI panels
- Databases and Data Access Libraries
- Calibration Scenarios

Crate Controller (1)

Run Control Hierarchy

- Online RC framework implements a concurrent hierarchical state model (CHSM)
- User actions executed on transition between states
- Parent controllers complete their transitions before initiating state transitions of their children

Requirements

- Conform to ATLAS run control system
- Initialise all our production and test modules, using various databases
- Handle final system and various subsets
- Perform physics and calibration/test runs
- Change between run types
- Monitor crate/module status
- Easy to implement new run types

Crate Controller (2)

Proposed Setup

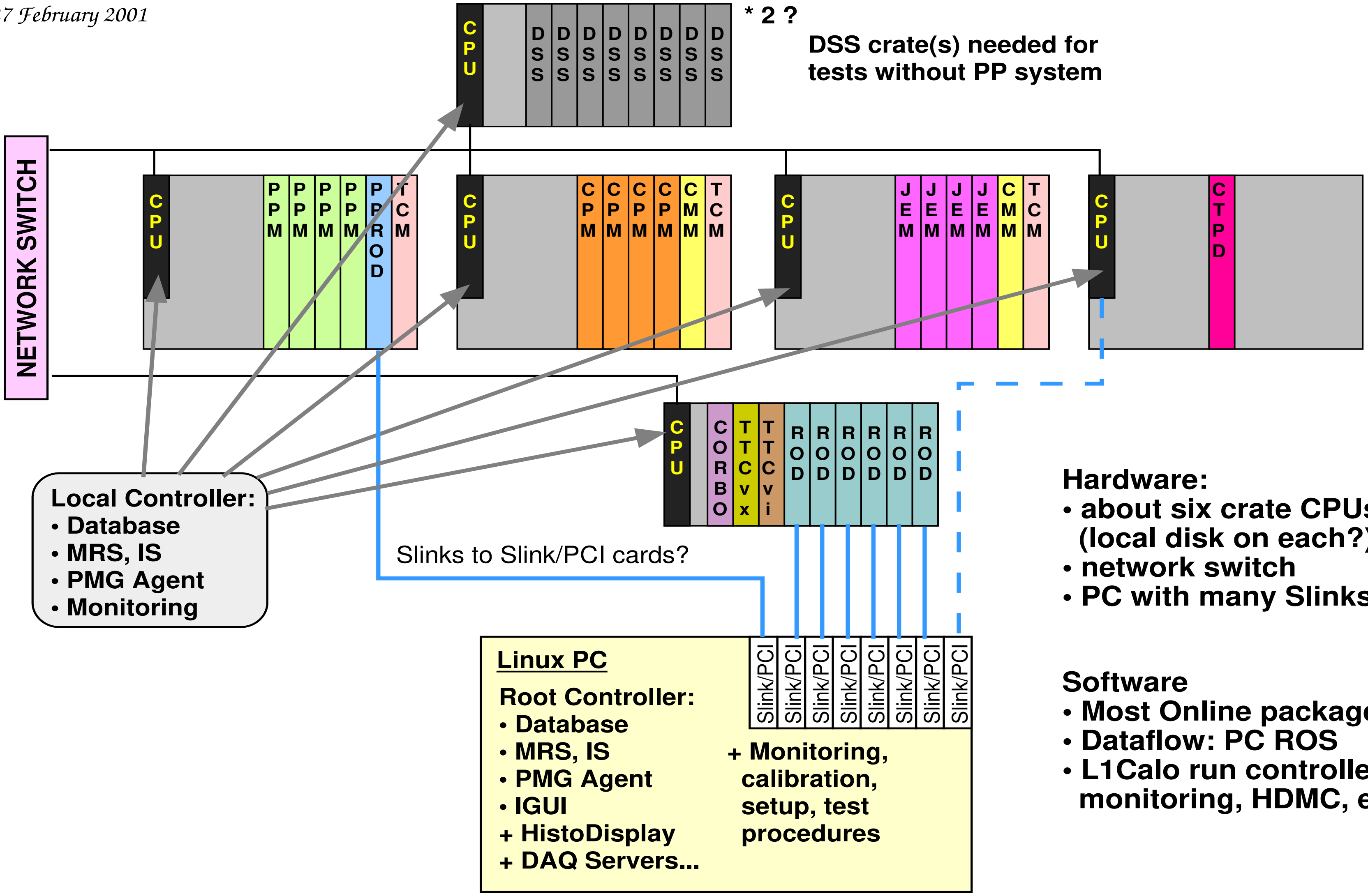
- “Local Controller” for each crate, running on the crate CPU
- Online Software services (eg DB access) required in all crates
- CPUs may be diskless (boot from L1Calo server) or may have local disks. [To be decided: question of data/software distribution vs network performance]

Controller Responsibilities

- Read run parameters from IS
- Load hardware configuration and other databases (some may depend on run type?)
- Implement actions for each state transition
- Separate thread/process for monitoring modules in the crate (eg link errors, PP rates?)
- Update crate status in IS

L1 Calo Trigger: Slice Test DAQ Setup

Murrough Landon
27 February 2001



Hardware:
- about six crate CPUs (local disk on each?)
- network switch
- PC with many Slinks

Software
- Most Online packages
- Dataflow: PC ROS
- L1Calo run controllers, monitoring, HDMC, etc

Crate Controller (3)

Implementation

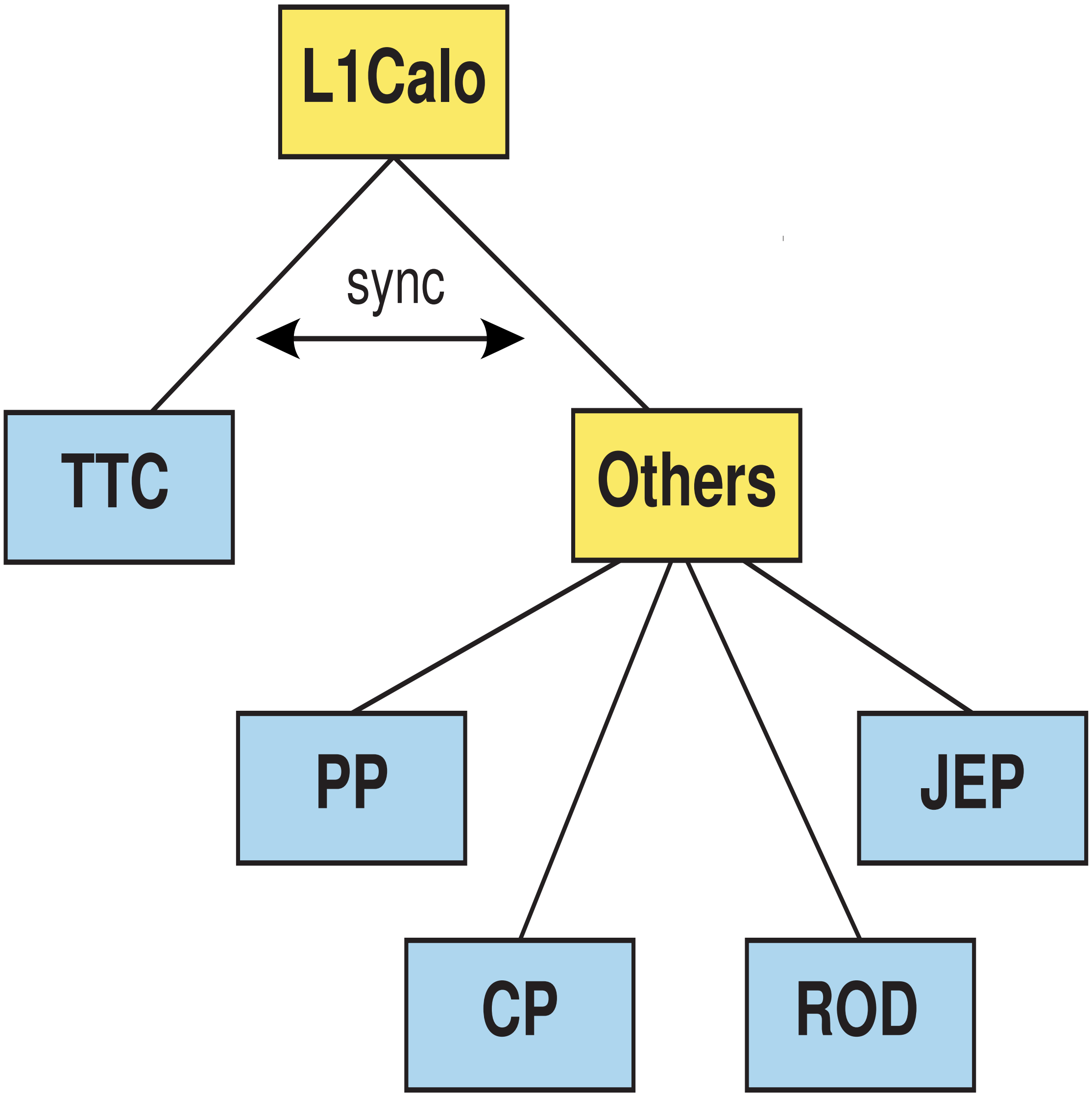
- Design a "CtrlModule" subclass for each module type
- CtrlModule provides methods for high level run control actions for the module
- Crate controller uses H/W database to create list of CtrlModules in its crate
- Crate controller is also the interface between other databases (trigger menu, calibration) and the CtrlModules
- Crate controller calls sets of CtrlModule methods to execute the various transition actions (may depend on run type)
- CtrlModule may own/link to HDMC Part(s) which will actually perform the actions
- CtrlModule responsible for updating its status in IS
- Considering best way to implement different run types

Crate Controller (4)

Synchronisation

- Run controllers act independently...
- ...but parents can control order of state transitions of their children
- Important to know about synchronisation requirements to design the RC hierarchy
- Likely controller hierarchy includes additional “synchronisation” controller to ensure TTC actions happen before/after other actions
- Also: do we need synchronisation/ordering within one crate?
- ...ie can crate controller just use a list of generic CtrlModules (or does it need to know which is which)?
- Guess CP/JEP: yes, TTC: no, ROD: yes (if no BUSY), PP: not sure (pipeline bus?)

Run Controller Hierarchy for L1Calo Slice Tests



Final system probably similar, but with more than one instance of PP, CP, JEP and ROD crates

Crate Controller (5)

Work so far...

- Draft document: simple requirements, detailed list of actions on each transition for each module
  `http://www.hep.ph.qmw.ac.uk/~landon/l1soft/docs`
  `RAL CVS: l1calo/doc`
- H/W configuration database created for slice test configuration
- ...many bugs/features in database editor reported
- Started prototype (mostly empty) crate controller
  `RAL CVS: l1calo/rc`
- ...presently just reads and reports the crate configuration from the database
- Hierarchy of controllers tested on single and multiple nodes
- Started implementing trigger menu schema and DAL
  `RAL CVS: l1calo/database, l1calo/confdb`

Crate Controller (6)

Next steps...

- Complete, circulate and review documents
- Extend Online H/W configuration database schema: would like our own module classes
- Complete (level 1 calo) trigger menu: schema, DAL, editor
- Implement similar for (at least some) calibration data
- NB Online confdb_gui will be extensible in version 0.0.14 (summer)
- Create IS classes for exchange of run parameters and module status information
- Provide java panels in the IGUI to set/display this information
- NB new IS java API available in Online S/W version 0.0.13 (imminent)
- Need to decide on and develop suitable approach to using HDMC as hardware access library.

Database (1)

Overview

- Three main categories:
- Hardware (and software) configuration: extension of Online s/w configuration database
- Trigger menu: physics choices. Defined offline, we need access to objects encapsulating level 1 trigger settings
- Calibrations: various sets of data...
- ...some derived online: must be stored offline
- ...some derived offline: must be accessed online

Hardware Configuration

- Present tests use generic DB Module class
- Actual module type derived from object ID (string)
- Preferable to have our own Module subclasses
- Can then add extra attributes: numbering, eta,phi mapping, links to FPGA versions, etc

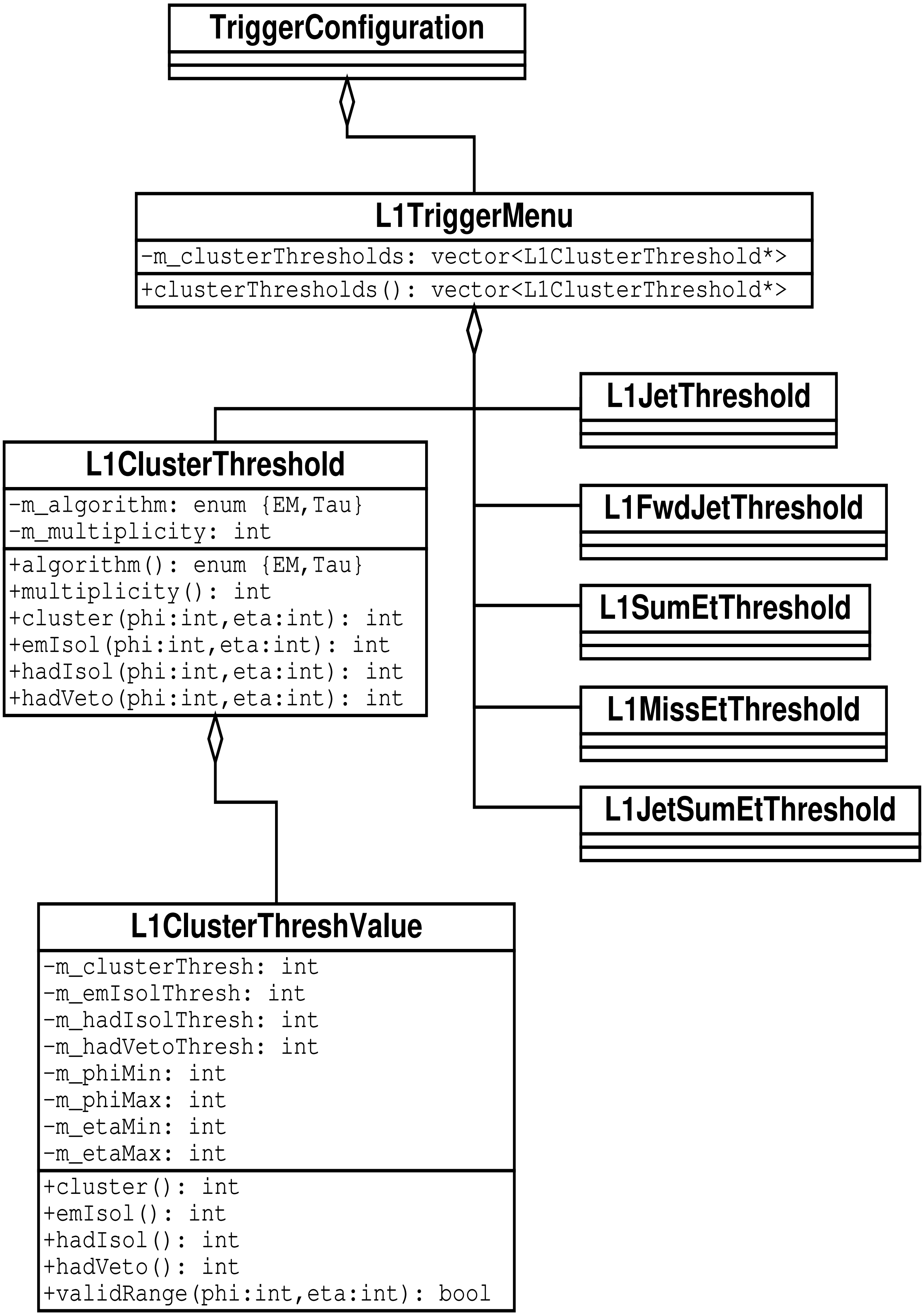
Database (2)

Trigger Menu

- Complete trigger description at all trigger levels is required eventually
- L1Calo software should see same API online and offline
- Online may be implemented as OKS objects?
- For slice tests we need something: OKS based schema
- New simulation also needs something soon

Calibrations

- Is OO database the right choice for some/all calibration data?


TriggerConfiguration
L1TriggerMenu
-m_clusterThresholds: vector<L1ClusterThreshold*>
+clusterThresholds(): vector<L1ClusterThreshold*>
L1JetThreshold
L1ClusterThreshold
-m_algorithm: enum {EM,Tau}
-m_multiplicity: int
+algorithm(): enum {EM,Tau}
+multiplicity(): int
+cluster(phi:int,eta:int): int
+emIsol(phi:int,eta:int): int
+hadIsol(phi:int,eta:int): int
+hadVeto(phi:int,eta:int): int
L1FwdJetThreshold
L1SumEtThreshold
L1MissEtThreshold
L1JetSumEtThreshold
L1ClusterThreshValue
-m_clusterThresh: int
-m_emIsolThresh: int
-m_hadIsolThresh: int
-m_hadVetoThresh: int
-m_phiMin: int
-m_phiMax: int
-m_etaMin: int
-m_etaMax: int
+cluster(): int
+emIsol(): int
+hadIsol(): int
+hadVeto(): int
+validRange(phi:int,eta:int): bool

Calibration Scenarios (1)

Single Run

- Use Pause/Resume transitions
- Start calibration sequencer process
- Take N triggers
- Issue Pause/Resume commands to Root controller
- Other controllers load next parameter value in the calibration sequence
- Iterate; stop run at end of sequence
- Pro: complete calibration in single file; analysed by single 100% monitor process;
- Con: controllers and monitor/analysis processes need to know the sequence
- TileCal do (some?) calibrations like this (except beam energy scans)

Calibration Scenarios (2)

Multiple Runs

- Each run uses single parameter value
- Start/stop runs, changing parameters
- Could all be controlled by a script
- Pro: less intelligence in crate controllers
- Con: need to process several files to extract one set of calibration values
- LAr may prefer this way: more like existing test beam habits

Our Calibration and Test Runs

- Inspired by TGC Use Case document, started detailing all our calibration, test, setup, check procedures
  `http://www.hep.ph.qmw.ac.uk/~landon/l1soft/docs`